Contents

Development and administration for Finance and Operations apps

Finance and Operations application documentation

Development and administration for Finance and Operations apps

What's new or changed

What’s new or changed in Finance and Operations apps

What’s new or changed in Platform updates

What's new or changed in Platform updates home page

Platform updates for version 10.0.23

Platform updates for version 10.0.22 (November 2021)

Platform updates for version 10.0.21 (October 2021)

Removed or deprecated features

Removed or deprecated features home page

Removed or deprecated platform features

Removed or deprecated features in previous releases

AX 2012 features that were postponed

End of mainstream support for Microsoft Dynamics AX 2009 and 2012

Continuous delivery

Continuous delivery home page

Development and continuous delivery FAQ

Exclude test packages from build output

Manage third-party models and runtime packages using source control

All-in-one deployable packages

Update model versions in the automated build

Data integration

Integration between Finance and Operations apps and third-party services

Priority-based throttling

Priority-based throttling FAQ

Data integration APIs
Data management package REST API
Service endpoints
   Service endpoints overview
   Service authentication troubleshooting
OData
Custom service development
Recurring integrations
Test services using third-party utilities
Finance and Operations Connector
Development for integration
   Data integration using data entities
      Data integration using data entities overview
      Develop an entity for data migration
      Develop composite data entities
      Configure financial cross-company data sharing
      Create record templates to facilitate data entry
      Create records using record templates
Data Integrator
   Configure a prospect to cash scenario
   Configure integration with Field Service
   Configure Project Service Automation integration
Consuming external web services
Electronic messaging
Data management
   Data management overview
Data entities
   Data entities overview
Configuration data projects
Copy configuration data between companies or legal entities
   Copy configuration data overview
   Configuration data packages
   Configuration data packages (July 2017 release only)
Configuration data templates
Track changes to an entity
Find information about standard data entities
Data templates with multiple worksheets
Enable change tracking for entities
Configuration keys and data entities
Configuration data packages
Data import and export jobs
  Data import and export jobs overview
  Synchronize date and time in import jobs
  Importing vouchers using the General journal entity
  Optimizing data migration for Finance and Operations apps
Development for data entities
  Design principles and best practices for data entities
  Create new data entities
  Data entity properties
  Create computed columns and virtual fields
  Cross-company behavior
  Handling country/region codes
  Inheritance patterns
  Data entity wizard rules
  Metadata properties
  Validations, defaults, and unmapped fields
  Security and data entities
  Use third-party service testing utilities
  Create read-only entities that expose financial dimensions
Bring your own database
Automated Entity store refresh
Data task automation
Data validation checklist workspace
Data management error descriptions
Azure Data Lake
Azure Data Lake overview
Configure export to Azure Data Lake
Finance and Operations apps in Azure Data Lake
Make entity store available as a Data Lake
Change data in Azure Data Lake

Database movement operations
Overview
Database movement toolkit
Quick start guides
Refresh database
Export a database
Import a database
Database point-in-time restore
Point-in-time restore of the production database to a sandbox environment
Enable just-in-time database access

Tutorials
Refresh for training purposes
Debug a copy of the production database
Export a copy of the standard user acceptance test (UAT) database
Golden configuration promotion
Destructive testing

RESTful API
Overview
Versioning and support
Authentication
Throttling

Reference
API v1 reference
List database backups
Create database refresh
Create a database export
Get operation activity status
Demo data
  Demo data overview
  Generate demo data using data packages

Deployment
  Deploy demo environments

Cloud deployment
  Cloud deployment overview
  Self-service deployment
    Self-service deployment overview
    Deploy a new environment
  Deployment FAQ
  Known issues with self-service deployment
  Maintenance operations for deployments
  Update an environment
  Troubleshoot environments deployed through self-service deployment
  Planned maintenance in self-service environments FAQ
  Business continuity and disaster recovery

Complete the Azure Resource Manager onboarding process

Complete the Azure Resource Manager onboarding process for US government

LCS projects
  Azure ExpressRoute and Finance and Operations
  Finance and Operations architecture
  Apply updates to cloud environments
  Apply updates and extensions to Commerce Scale Unit (cloud)
  Auto-update for Commerce Scale Unit (cloud)
  Initialize Commerce Scale Unit (cloud)
  Migrate channels to a different Commerce Scale Unit
  Retail data residency

On-premises deployment
  On-premises deployment home page
  Prepare for on-premises deployments
    On-premises deployment overview
    Plan and prepare for on-premises deployments
Microsoft Dynamics 365 Finance + Operations (on-premises) supported software

Hardware sizing requirements

Authentication in Dynamics 365 Finance + Operations (on-premises)

Set up and deploy on-premises environments

Set up on-premises projects in LCS

Set up and deploy on-premises environments (Platform update 41 and later)

Set up and deploy on-premises environments (Platform updates 12 through 40)

Installation steps for Retail channel components in an on-premises environment

Develop and deploy custom models to on-premises environments

Environment operations

Apply updates to an on-premises deployment

Certificate rotation

Client internet connectivity

Configure Batch-only and Interactive-only AOS nodes in on-premises deployments

Configure high availability for SQL Server Reporting Services (SSRS) nodes

Configure proxies for on-premises environments

On-premises diagnostics

On-premises disaster recovery configuration

PowerBI.com integration with on-premises environments

Reconfigure environments

Redeploy on-premises deployments

Remove and reinstall, or add an AOS node

Reuse the same AD FS instance for multiple environments

Local agent

AD FS Microsoft 365 compatibility

Local agent pre-deployment and post-deployment scripts

Deployment configurations for the local agent

Update the local agent

Troubleshooting

Troubleshoot on-premises deployments
Scripts for resolving issues in on-premises environments

Deploy custom code
Create deployable packages of models
Install deployable packages from the command line
Uninstall a package
Troubleshoot package application issues

Deploy custom help
Custom help overview
Preparing content for use with the Help pane
Deploying custom help to Azure
Connect your custom help to the Help pane
Custom help toolkit
  Overview
  The HTML From Repo Generator tool
  The Convert HTML To JSON tool
  The HTML Locale Changer tool
  Convert Dynamics AX custom help for use in Dynamics 365
Language and locale descriptors in across product and help
Extend, customize, and collaborate on the help

Deployment considerations
Sovereign and local cloud deployment options for Dynamics 365 Finance and Dynamics 365 Supply Chain Management
Finance and Supply Chain Management in US Government Community Cloud (GCC)
Finance and Supply Chain Management operated by 21Vianet in China
Finance and Operations apps in France

Develop and customize
Develop and customize home page
Application stack and server architecture
Get evaluation copies
Sign up for preview subscriptions
Deploy and access development environments
Development environments
Configure one-box development environments
Development and build VMs that don’t allow admin access FAQ
Rename local environments to access Azure DevOps
Development system requirements
Version control, metadata search, and navigation
Build automation using Azure
  Build automation using Microsoft-hosted agents and Azure Pipelines
  Add license files to a deployable package in Azure Pipelines
  Create deployable packages in Azure Pipelines
  X++ model-versioning in Azure Pipelines
  Download assets by using Azure Pipelines
  Upload assets by using Azure Pipelines
  Deploy assets by using Azure Pipelines
  Create a Lifecycle Services (LCS) connection in Azure Pipelines
  Update the hosted Azure Pipeline for new NuGet packages
  Update a legacy pipeline in Azure Pipelines
Fleet Management sample application
  End-to-end scenario for the Fleet Management sample application
Visual Studio tools
  Development tools
  Development tools tutorial
  Application checker
  Application Explorer
  Build and debug projects
Build operations
Code editor features
Create models and data model elements
  Create models and data model elements overview
  Naming guidelines for extensions
  Turn off model customization and deprecate functionality
Customization Analysis Report
Element designers
Commands for determining element use
Export and import models
Metadata search in Visual Studio
Models and packages
Finance and Operations project type
Tools add-ins for Visual Studio
Update the Visual Studio development tools
X++ programming language
X++ language support
   Visual Studio requirements for X++
   Debug X++ code
   EventHandlerResult classes in request or response scenarios
   Write business logic using C# and X++ source code
   LINQ provider for C#
   Write best practice rules
   Application Explorer properties
   SysSetupConfigAttribute attribute
X++ language reference
   X++ language reference overview
   Variables and data types
   Statements, loops, and exception handling
   SQL connection error X++ exception
   X++ operators
   X++ operator precedence
   X++ Classes and methods
   X++ data selection and manipulation
   X++ macros
   X++ attribute classes
   X++ and C# comparison
   X++ syntax
API, class, and table reference
API, class, and table resources
X++ compile-time functions
X++ runtime functions
  X++ runtime function resources
  X++ business runtime functions
  X++ container runtime functions
  X++ conversion runtime functions
  X++ date runtime functions
  X++ math runtime functions
  X++ reflection runtime functions
  X++ session runtime functions
  X++ string runtime functions
System tables
Extensibility
  Extensibility home page
Introduction and getting started
  Application extensibility plan
  Extensibility requests
  Extensibility FAQ
Migrate from overlayering to extensions
  Customize model elements through extension (tutorial)
  Customize through extension and overlayering
What's new
  What's new or changed for extensibility
    Extensibility changes version 10.0.3
    Extensibility changes version 10.0.2
    Extensibility changes version 10.0.1
    Extensibility changes version 10.0
    Extensibility changes version 8.1.3
    Extensibility changes version 8.1.2
    Extensibility changes version 8.1.1
    Extensibility changes version 8.1
Extensibility changes version 8.0.4
Extensibility changes version 8.0.3
Extensibility changes version 8.0.2
Extensibility changes version 8.0.1
Extensibility changes in version 8.0
Extensibility changes in version 7.3
Extensibility changes July 2017

Fundamentals

Intrusive customizations
Class extension model
Class extension via method wrapping and Chain of Command
Naming guidelines for extensions
Relax model restrictions to refactor overlayering into extensions

Creating extensions

Add values to enums
Modify extended data types
Register subclasses for factory methods
Respond using EventHandlerResult
Extend the RunBase class
Customize application startup using delegates
Modify existing fields in a table
Add fields to tables
Add indexes to tables
Add relations to tables
Modify table properties
Add methods to tables
Perform business actions throughout the lifecycle of table records
Add data sources to forms
Change form captions
Modify form control properties
Extend the scope of number sequences
Add new inventory dimensions
Price and discount extensibility
Table map extension
Extend table maps used as interfaces
Extend table maps used for versioning
Extending decimal point precision for selected data types

Creating extensible solutions
Write extensible code
Classes
Methods
Forms
Extended data types
Extensible enums
Delegates
Tables
Extensibility attributes for methods
Breaking changes
Compatibility checker tool

Performance
Take traces using Trace parser
Single-user testing using Performance SDK and Task recorder
Multi-user testing using Performance SDK
Troubleshooting guide for testing with the Performance SDK
Performance SDK and multiuser testing in on-premises environments
Diagnose issues and analyze performance using Trace parser
Performance timer

Testing support in Visual Studio
Testing and validations
Test projects in Visual Studio
Deploy and use a continuous build and test automation environment
SysTest filtering using class and method attributes
Acceptance test library
Acceptance test library resources
Navigation concepts
Test data methods
Entities in the acceptance testing library
Acceptance test library commands
Creators in the acceptance test library
Queries in the acceptance test library
Specification classes
Acceptance test library code generation wizard
Best practices for the acceptance test library
Frequently asked questions
Date effectivity
Development for independent software vendors
 ISV development home page
 Link X++ modules to packages by using ISV Studio
 ISV licensing
 ISV licensing on-premises
General Data Protection Regulation (GDPR)
 General Data Protection Regulation overview
 Respond to GDPR data requests
  Respond to GDPR data requests resources
 Asset classifications
 Person search report
 Extend the Person search report
 Manage access to sensitive data
 Respond to requests for personal data using Human Resources
 Respond to requests for personal data in AX 2012
 GDPR data requests for LCS
Lifecycle Services
 Resources for Lifecycle Services
  Lifecycle Services resources
 LCS for customers
 LCS for partners
RESTful APIs
  Start and stop environments
  Fetch environment metadata
  Fetch environment history
  Fetch environment RSAT certificate ZIP
What's new or changed in Lifecycle Services
Lifecycle Services user guide
Project onboarding
Subscription estimator
Configure security
Issue search
Configuration manager
  Configuration manager overview
  Set up Configuration manager
Configure the code upgrade service
Create or update methodologies
Business process modeler (BPM)
  Business process modeler (BPM) overview
  Business process libraries in Business process modeler
  Create, edit, and browse Business process modeler libraries
  Complete tasks in Business process modeler
  Work with activity diagrams in BPM libraries
  Synchronize BPM libraries with Azure DevOps
  Create user acceptance test libraries using task guides and BPM
  Flowcharts in Business process modeler
  Upload custom business processes to Business process modeler
Service requests
  Cloud operations and servicing
  Submit service requests to the Dynamics Service Engineering team
  Planned maintenance window FAQ
Monitoring and diagnostics
  Monitoring and diagnostics tools
Use Azure Data Explorer to query raw information logs
Restart environment services
Report a production outage
Track user sign-ins
Asset library
Performance troubleshooting using tools in LCS
Query cookbook
Expired subscriptions and data deletion
Microsoft Dynamics 365 Translation Service
  Dynamics 365 Translation Service overview
  Translate user interface files
  Translate documentation files
  Translation memory files
Microsoft Power Platform integration
  Microsoft Power Platform integration with Finance and Operations
  Enabling the Power Platform integration
  Authentication and authorization
Dual-write
  Dual-write home page
  What's new or changed in dual-write
  Dual-write overview
  Dual-write setup
    System requirements for dual-write
    Guidance for how to set up dual-write
    Considerations for initial synchronization
    Limits for live synchronization
    Dual-write setup from Lifecycle Services
    Enable dual-write for existing Finance and Operations apps
    Currency data-type migration for dual-write
    Set up the mapping for the sales order status fields
    Filter intercompany orders to avoid synchronizing Orders and OrderLines
Managing dual-write
Customize entity and field mappings
Customization guidance for dual-write
Handling multiple entity maps
Edit a legal entity after dual-write setup
Pause dual-write for maintenance
Error management and alert notifications
Application lifecycle management
User-specified team owner
Unlink and relink dual-write environments
Mapping concepts between apps
Integrated customer master
Vendor integration
Customer loyalty cards and reward points
Product integration
Company integration
Organization hierarchy on Dataverse
Finance and tax data
Integrate procurement in Supply Chain Management with Field Service
Sync on-demand with the Supply Chain Management pricing engine
Prospect to cash in dual-write
Use Dynamics 365 Commerce pricing engine with Dynamics 365 Sales
In-house assets for servicing
Inventory availability
Integrated worker, job, and position
Party and global address book
Using Microsoft Power Apps portals with the Party data model
Upgrade to the party and global address book model
Note integration
Mapping reference
Support for dual-write
Support for Field Service and Project Service Automation solutions
Migrate Prospect to cash data from Data Integrator to dual-write
Add-ins
   Add-ins overview
Policies and communications
Mobile platform
Mobile platform resources
Get started
   Get started with the mobile platform
   Architecture and design considerations
   Business logic events
   Page design guidelines
   Action design guidelines
   Form design requirements
Common tasks
   Configure workspaces using the SysAppWorkspace class
   Make fields on mobile app pages clickable
   Show counts in fields
   Help secure mobile workspaces
   Localize mobile workspaces
   Make fields mandatory using workspace classes
Server-side development reference
   Server-side development (workspace X++ APIs)
Client-side development reference
   Client-side design APIs
Client APIs
   Client APIs home page
   PageState enumeration
   Application module
   Defer module
   Event module
   Control module
   Container module
   Field module
File Uploader module
Group module
Hyperlink module
Image module
Input module
List module
Lookup module
Multi-Lookup module
Page module
Pagelink module
Part module
Services module
Value module
Application type
ApplicationMetadata type
AsyncService type
CacheService type
CompleteEventArgs type
ContainerControl type
ContainerControlDesign type
ContainerControlMetadata type
ControlMetadata type
DataService type
Deferred type<T>
Design type
EventHook type<T>
Field type
FieldDesign type
FieldMetadata type
FileUploader type
FileUploaderDesign type
FileUploaderMetadata type
GenericValue type
getDataSource type
Group type
GroupDesign type
GroupMetadata type
HyperLink type
HyperLinkDesign type
HyperLinkMetadata type
Image type
ImageDesign type
ImageMetadata type
InputControl type
InputControlDesign type
InputControlMetadata type
List type
ListDesign type
ListMetadata type
Lookup type
LookupDesign type
LookupMetadata type
MetadataService type
MultiLookup type
MultiLookupDesign type
MultiLookupMetadata type
NavigationArgs type
NumberSequenceConfig type
Page type
PageData type
PageLink type
PageLinkDesign type
PageLinkMetadata type
PageMetadata type
Process automation framework
  Process automation framework development
  Type registration
  Series registration
  Process parameters
  User-configurable queries
  Run the process
  Log results and messages
  Customize the user interface
Regression suite automation tool
  Regression suite automation tool overview
  Installation and configuration
  Use the Regression suite automation tool
  Maintain test cases within Regression suite automation tool
  Validate expected values
  Chain test cases
  Derived test cases
  Upgrade the parameter files
Regression suite automation tool tutorial
  Configure non-administrator users
  Best practices
Data agnostic testing using the Regression Suite Automation Tool
Troubleshooting

Release solutions using Lifecycle Services
- Develop and release
- Stage and publish
- Add methodologies
- Set up Business process modeler libraries
- Migrate code
- Validate applications
- Process and consume data packages
- Back up solution databases

Localization and regulatory features
- Globalization resources
- Classification of localization features
- Apply country/region context
- Regulatory certification information in feature titles
- Regulatory updates
- Separation of localization models
- Submit regulatory alerts

Support
- Get support for Finance and Operations and Lifecycle Services
- Dynamics 365 support overview
- Get support for Finance and Operations operated by 21Vianet in China
- Set up technical support for Finance and Operations apps
- Manage support experiences for Finance and Operations apps
- Production support and monitoring

System administration
- System administration home page
- Add links to your legal terms and privacy statement
- License codes and configuration keys report
- Cross-company data sharing
- Maintenance mode
- Preconfigured system accounts
Export B2B users to Azure AD

Data maintenance

Security
  Role-based security
  Security architecture
  Encryption in Finance and Operations apps
  Set the session inactivity timeout
  Out-of-box security reports
  Create new users
  User session management
  Block access by location with Azure AD Conditional Access
  Import users in bulk
  Set up segregation of duties
  Identify and resolve conflicts in segregation of duties
  Assign users to security roles
  Import or export customized security configuration using Data management
  Security diagnostics for task recordings
  Extensible data security policies
  Create a security policy
  Stay compliant with user licensing requirements
  View independent software vendor license status

Process automation

Batch processing
  Batch processing overview
  Batch processing and batch servers
  Create batch jobs
  Enabling automatic retries on batch jobs
  Copy a batch job
  Active batch periods
  Daylight saving time support for active batch periods
  Batch manager security role
  Configure batch alerts
Enhanced batch forms
Clean up the batch history
Cancel an executing batch job
Batch OData API
Optimization advisor
  Optimization advisor overview
  Create rules for Optimization advisor
Report a production outage
Configure and manage database logging
Build OData metadata cache when the AOS starts
Setting Up Azure Key Vault client
Cleanup routines
Upgrades, updates, and hotfixes
  Upgrades, updates, and hotfixes resources
Upgrade from AX 2012
  Upgrade from AX 2012 to Finance and Operations
  AX 2012 upgrade - Plan using the Upgrade analyzer tool
  AX 2012 upgrade – Estimate effort using the Code upgrade service
  AX 2012 upgrade - Deploy a demo environment for analysis
  AX 2012 upgrade - Data upgrade in development environments
  AX 2012 upgrade - Pre-upgrade checklist for data upgrade
Upgrade from AX 2012 - SQL Transactional Replication
Upgrade from AX 2012 - Data upgrade in self-service environments
  AX 2012 upgrade - On-premises environments
  AX 2012 upgrade - Cutover testing
  AX 2012 upgrade - Post-upgrade tasks
  AX 2012 upgrade - Functional test passes
  AX 2012 upgrade - Prepare for go-live
  AX 2012 upgrade - Go live
Upgrade from AX 2012 - Data upgrade FAQ
Changes that affect upgrade from AX 2012
  Make the chart of accounts delimiter unique
Project resource scheduling data model
Workflow subsystem changes
Aggregate models used instead of cubes for analytics
Migrate upgraded sales cubes to the entity store
Upgrade budget planning

Migrate data from AX 2009
Migrate data from AX 2009 to Finance and Operations
AX 2009 migration - Install the Data migration tool
AX 2009 migration - Generate maps
AX 2009 migration - Create templates
AX 2009 migration - Create migration groups
AX 2009 migration - Export packages
AX 2009 migration - Import packages

Code migration and upgrade
Code migration and upgrade home page
Prepare to migrate code
Configure the Azure DevOps mapping
Model split
Solve dependencies among models using delegates

Upgrade to a recent update (cloud)
Process for moving to the latest update
Paths to One Version
  Self-service upgrade to the latest version
  Rebuild and update to the latest version
Software lifecycle policy and cloud releases
Apply the latest platform update to environments
Upgrade data in development, or demo environments
Update development tools
Plan and prepare for compiling code against the latest update

Upgrade to a recent update (on-premises)
Apply updates to on-premises deployments
Redeploy an on-premises deployment
In-place upgrade process for on-premises environments

Hotfixes

Get updates from Lifecycle Services
Apply updates to cloud environments
Install metadata hotfixes in development environments
Patch Reporting Services in one-box environments
Update Visual Studio development tools

Deprecations

Removed or deprecated features
Deprecation of methods and metadata elements
Deprecated APIs

User interface development

User interface development home page

Tutorials

Build the Rental Charge Type form
Build the Customer form
Build navigation
Modify a workspace

Forms

Navigation concepts
Page layout in the web client
Dynamics Symbol font
Saved views
Build forms that fully utilize saved views
Test forms that use custom patterns
Create shareable, secured URLs (deep links)
Accessibility in forms, products, and controls
Customize field descriptions

Controls

Action controls
Input controls and grid column sizes
Check box support in tree controls
Filtering options
Power Apps Host control
Window management
  Migrate context menu code
  Migrate mouse double-click logic
  Contextual data entry for lookups
  HierarchyViewer control
Lookup controls
  File upload control
System-defined buttons
Images on a page or in a grid
  Font and background colors for input, table, and grid controls
Right-to-left language support and bidirectional text
Create icons for workspace tiles
Public JavaScript APIs for extensible controls
Control checklist
Messages
  Feature callouts
Slider and MessageBox dialogs
  Message center, message bar, and message details API
  Message center, message bar, and message details FAQ
Form pattern guidelines
  Form patterns for migrated forms
  Form styles and patterns
  Visual Studio add-ins supporting form patterns
Form patterns
  General form guidelines
Details Master form pattern
Details Transaction form pattern
Form Part Section List form patterns
List Page form pattern
Simple Details form pattern
Simple List and Details form pattern
Simple List form pattern
Table of Contents form pattern
Task Double form pattern
Task Single form pattern
Wizard form pattern
Workspace form pattern
Secondary form patterns
Advanced selection form pattern
Dialog form pattern
Drop Dialog form pattern
Lookup form pattern
FactBox form patterns
Sub patterns
Custom Filter Group subpattern
Dimension Entry Control subpattern
Dimension Expression Builder subpattern
Fields and Field Groups subpattern
Filters and Toolbar subpatterns
Fill Text subpattern
Horizontal Fields and Buttons Group subpattern
Image Preview subpattern
List Panel subpattern
Nested Simple List and Details subpattern
Section Chart form pattern
Section Power BI subpattern
Section Related Links subpattern
Section Stacked Chart subpattern
Section Tabbed List subpattern
Section Tiles subpattern
Tabular Fields subpattern
Toolbar and Fields subpattern
ToolBar and List subpattern
Workspace Page Filter Group subpattern
Control extensibility
  Build extensible controls
  Keyboard shortcuts for extensible controls
  Extensible control programming reference
  Control extensibility
    Create localizable labels
    Extensible control layout guidelines
    Control the text that Task Recorder generates for a control
Build workspaces
  Build operational workspaces
  Tile and list caching for workspaces
Task Recorder
  Task Recorder resources
  Task Recorder quick reference
    Create documentation or training by task recordings

Dynamics 365 Supply Chain Management
Dynamics 365 Finance
Dynamics 365 Commerce
Learn how to make Finance and Operations applications work for your business, using the resources in this topic to find great content for end users, developers, and IT professionals.

Much of this content also applies to the related products: Dynamics 365 Commerce and Dynamics 365 Human Resources.

Before you buy
- Sign up for a preview subscription
- Choose a deployment option
- Buy Finance and Operations (on-premises)

Implementation lifecycle
- FastTrack for Dynamics 365
- Onboarding a project
- Preparing for go-live

One Version service updates
- One Version service updates FAQ
- Software lifecycle policy: Cloud
- Software lifecycle policy: On-premises
- Standard and First release service updates
- What's new or changed
- Upgrades, updates, and hotfixes
- Apply updates to cloud environments
- Configure service updates
- Pause service updates
- Get notified about service updates
- Data task automation
- Regression Suite Automation tool

Integrations
- Business events
- Data entities
- Integration using Microsoft Power Automate

Financial management
- Accounts payable
- Accounts receivable

Supply chain management
- Cost management
- Inventory management
- Master planning
- Procurement and sourcing
- Product information management
- Production control
- Sales and marketing
- Transportation management
- Warehouse management

Intelligence
- Analytics
- Business documents
- Financial reporting
- Regulatory reporting

Development
- Extensibility
- Office integration
- Continuous delivery
- Mobile platform
- Demo data

Administration
- Cloud deployment
- On-premises deployment
- Upgrade
- Servicing
- Lifecycle Services
- Organization administration

Related products
- Dynamics 365 Commerce
- Call center
- Channel setup and management
- MPOS and Cloud POS
- Commerce developer and administration
- Dynamics 365 Human Resources
- Administrator Guide
- Developer Guide
- User Guide
<table>
<thead>
<tr>
<th>Human resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benefits</td>
</tr>
<tr>
<td>Employee development and training</td>
</tr>
<tr>
<td>Questionnaires</td>
</tr>
<tr>
<td>Recruiting</td>
</tr>
<tr>
<td>US payroll</td>
</tr>
</tbody>
</table>
Development and administration for Finance and Operations apps includes:

- Administrator experience and Lifecycle Services
- Developer experience
- Intelligence
- Mobile apps
- Data management and data entities
- Office integration

Developer experience

The developer experience is based on modern tooling using Visual Studio and .NET components.

- The development tools are decoupled from any running environment, which means that you develop against local, XML-based files, not the online database.
- Microsoft Visual Studio is the development environment. Finance and Operations customizes the Visual Studio environment to provide you with a smooth and familiar experience.
- The X++ compiler generates Common Intermediate Language (CIL) for all features. CIL is the same intermediate language used by other .NET-based (managed) languages, such as the C# programming language.
- You can leverage the browser-based client and design patterns for forms to provide an improved end-user experience.
- The Application Lifecycle Management (ALM) system supports build automation, test automation, and deployment of models to the cloud.

For more information, see Develop and customize home page.

Administrator experience and Lifecycle Services

Finance, Supply Chain Management, and Commerce are cloud-hosted. As an IT professional, you can use Dynamics Lifecycle Services (LCS) to monitor and tune your environments, deploy features, and stay up to date with recent hotfixes. Within your deployment, you can configure security, and manage when processes run. You can also use LCS when you are called on to support business intelligence and reporting, mobile apps, Office, and other integrations.

BI & reporting

Finance and Operations provides five distinct reporting experiences. Specialized tools are provided to meet the complex and diverse reporting needs of various functions throughout the organization.

- Operational views – Designed to address the specific needs of a given business persona.
- Business documents – Static documents used to capture and exchange processed business data.
- Analytical tools and visualizations – Personalized presentations of logical calculations that allow the user to explore their data.
Mobile apps
The Finance and Operations mobile app empowers your organization to mobilize its business processes. After your IT admin enables the mobile workspaces feature for your organization, users can sign in to the app and immediately begin to run business processes from their mobile devices. The Dynamics 365 for Finance and Operations mobile app includes the following features that can help increase productivity:

- Users can view, edit, and act on business data, even if they have intermittent network connectivity or their mobile devices are offline. When a device reestablishes a network connection, offline data operations are automatically synchronized with Finance and Operations.
- IT admins or developers can build and publish mobile workspaces that have been tailored to their organization. The app uses your existing code assets, so you don’t have to re-implement your validation procedures, business logic, or security configuration.
- IT admins or developers can easily design mobile workspaces by using the point-and-click workspace designer that is included with the Finance and Operations web client.
- IT admins or developers can optionally optimize the offline capabilities of workspaces by using the Business logic extensibility framework. Because data continues to be processed while a device is offline, your mobile scenarios remain rich and fluid, even if devices don’t have constant network connectivity.

Data management and data entities
Data from Finance and Operations can easily be integrated with Microsoft and non-Microsoft data sources using Dataverse, Power Apps, and Power BI. For more information, see Data entities overview.

Office integration
The Microsoft Office integration capabilities provide users with a productive environment that helps them get the job done by using Office products. For more information, see Office integration overview.

eLearning courses
For online courses and training, check out Dynamics 365 Finance and Operations on Microsoft Learn.
Application releases

To see what's new or changed in each release of a Finance and Operations app, see the following topics:

- **Finance**: What's new or changed in Dynamics 365 Finance
- **Supply Chain Management**: What's new or changed in Dynamics 365 Supply Chain Management
- **Commerce**: What's new or changed in Dynamics 365 Commerce
- **Human Resources**: What's new or changed in Dynamics 365 Human Resources

Platform updates

To see what's new or changed in the Platform updates for Finance and Operations apps, see the following topic:

- What's new or changed in Platform updates

Lifecycle Services releases

To see what's new or changed in Lifecycle Services, see the following topic:

- What's new or changed in Lifecycle Services (LCS)
To see what’s new or changed in the Platform updates for Finance and Operations apps, see the following topics.

<table>
<thead>
<tr>
<th>VERSION</th>
<th>BUILD NUMBER</th>
<th>AUTO-UPDATE AVAILABILITY</th>
<th>LEARN MORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform updates for 10.0.23</td>
<td>7.0.6206</td>
<td>January 2022</td>
<td>Platform updates for version 10.0.23 of Finance and Operations apps</td>
</tr>
<tr>
<td>Platform updates for 10.0.22</td>
<td>7.0.6164</td>
<td>November 2021</td>
<td>Platform updates for version 10.0.22 of Finance and Operations apps</td>
</tr>
<tr>
<td>Platform updates for 10.0.21</td>
<td>7.0.6129</td>
<td>October 2021</td>
<td>Platform updates for version 10.0.21 of Finance and Operations apps</td>
</tr>
<tr>
<td>Platform updates for 10.0.20</td>
<td>7.0.6060</td>
<td>August 2021</td>
<td>Platform updates for version 10.0.20 of Finance and Operations apps</td>
</tr>
<tr>
<td>Platform updates for 10.0.19</td>
<td>7.0.6009</td>
<td>June 2021</td>
<td>Platform updates for version 10.0.19 of Finance and Operations apps</td>
</tr>
<tr>
<td>Platform updates for 10.0.18</td>
<td>7.0.5968</td>
<td>May 2021</td>
<td>Platform updates for version 10.0.18 of Finance and Operations apps</td>
</tr>
<tr>
<td>Platform updates for 10.0.17</td>
<td>7.0.5934</td>
<td>April 2021</td>
<td>Platform updates for version 10.0.17 of Finance and Operations apps</td>
</tr>
<tr>
<td>Platform updates for 10.0.16</td>
<td>7.0.5860</td>
<td>February 2021</td>
<td>Platform updates for version 10.0.16 of Finance and Operations apps</td>
</tr>
<tr>
<td>Platform updates for 10.0.15</td>
<td>7.0.5816</td>
<td>January 2021</td>
<td>Platform updates for version 10.0.15 of Finance and Operations apps</td>
</tr>
<tr>
<td>Platform updates for 10.0.14</td>
<td>7.0.5778</td>
<td>November 2020</td>
<td>Platform updates for version 10.0.14 of Finance and Operations apps</td>
</tr>
<tr>
<td>Platform updates for 10.0.13</td>
<td>7.0.5746</td>
<td>October 2020</td>
<td>Platform updates for version 10.0.13 of Finance and Operations apps</td>
</tr>
<tr>
<td>VERSION</td>
<td>BUILD NUMBER</td>
<td>AUTO-UPDATE AVAILABILITY</td>
<td>LEARN MORE</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------</td>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Platform updates for 10.0.12</td>
<td>7.0.5688</td>
<td>August 2020</td>
<td>Platform updates for version 10.0.12 of Finance and Operations apps</td>
</tr>
<tr>
<td>Platform updates for 10.0.11</td>
<td>7.0.5644</td>
<td>July 2020</td>
<td>Platform updates for version 10.0.11 of Finance and Operations apps</td>
</tr>
<tr>
<td>Platform updates for 10.0.10</td>
<td>7.0.5600</td>
<td>May 2020</td>
<td>Platform updates for version 10.0.10 of Finance and Operations apps</td>
</tr>
<tr>
<td>Platform update 33</td>
<td>7.0.5559</td>
<td>April 2020</td>
<td>What's new or changed in Platform update 33 for Finance and Operations apps</td>
</tr>
<tr>
<td>Platform update 32</td>
<td>7.0.5493</td>
<td>February 2020</td>
<td>What's new or changed in Platform update 32 for Finance and Operations apps</td>
</tr>
<tr>
<td>Platform update 31</td>
<td>7.0.5457</td>
<td>January 2020</td>
<td>What's new or changed in Platform update 31 for Finance and Operations apps</td>
</tr>
<tr>
<td>Platform update 30</td>
<td>7.0.5407</td>
<td>November 2019</td>
<td>What's new or changed in Platform update 30 for Finance and Operations apps</td>
</tr>
<tr>
<td>Platform update 29</td>
<td>7.0.5372</td>
<td>October 2019</td>
<td>What's new or changed in Platform update 29 for Finance and Operations apps</td>
</tr>
<tr>
<td>Platform update 28</td>
<td>7.0.5314</td>
<td>July 2019</td>
<td>What's new or changed in Dynamics 365 for Finance and Operations platform update 28</td>
</tr>
<tr>
<td>Platform update 27</td>
<td>7.0.5286</td>
<td>June 2019</td>
<td>What's new or changed in Dynamics 365 for Finance and Operations platform update 27</td>
</tr>
<tr>
<td>Platform update 26</td>
<td>7.0.5257</td>
<td>May 2019</td>
<td>What's new or changed in Dynamics 365 for Finance and Operations platform update 26</td>
</tr>
<tr>
<td>Platform update 25</td>
<td>7.0.5222</td>
<td>April 2019</td>
<td>What's new or changed in Dynamics 365 for Finance and Operations platform update 25</td>
</tr>
<tr>
<td>VERSION</td>
<td>BUILD NUMBER</td>
<td>AUTO-UPDATE</td>
<td>AVAILABILITY</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Platform update 24</td>
<td>7.0.5179</td>
<td>March 2019</td>
<td></td>
</tr>
<tr>
<td>Platform update 23</td>
<td>7.0.5126</td>
<td>January 2019</td>
<td></td>
</tr>
<tr>
<td>Platform update 22</td>
<td>7.0.5095</td>
<td>December 2018</td>
<td></td>
</tr>
<tr>
<td>Platform update 21</td>
<td>7.0.5073</td>
<td>November 2018</td>
<td></td>
</tr>
<tr>
<td>Platform update 20</td>
<td>7.0.5030</td>
<td>September 2018</td>
<td></td>
</tr>
<tr>
<td>Platform update 15</td>
<td>7.0.4841</td>
<td>March 2018</td>
<td></td>
</tr>
<tr>
<td>Platform update 12</td>
<td>7.0.4709</td>
<td>November 2017</td>
<td></td>
</tr>
<tr>
<td>Platform update 11</td>
<td>7.0.4679.35176</td>
<td>October 2017</td>
<td></td>
</tr>
<tr>
<td>Platform update 10</td>
<td>7.0.4641.16233</td>
<td>August 2017</td>
<td></td>
</tr>
<tr>
<td>Platform update 9</td>
<td>7.0.4612.35162</td>
<td>July 2017</td>
<td></td>
</tr>
<tr>
<td>VERSION</td>
<td>BUILD NUMBER</td>
<td>AUTO-UPDATE AVAILABILITY</td>
<td>LEARN MORE</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------</td>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Platform update 8</td>
<td>7.0.4565.16212</td>
<td>June 2017</td>
<td>What's new or changed in Dynamics 365 for Finance and Operations, Enterprise edition platform update 8</td>
</tr>
<tr>
<td>Platform update 7</td>
<td>7.0.4542.16189</td>
<td>May 2017</td>
<td>What's new or changed in Dynamics 365 for Operations platform update 7</td>
</tr>
<tr>
<td>Platform update 6</td>
<td>7.0.4509.16180</td>
<td>April 2017</td>
<td>What's new or changed in Dynamics 365 for Operations platform update 6</td>
</tr>
<tr>
<td>Platform update 5</td>
<td>7.0.4475.16165</td>
<td>March 2017</td>
<td>What's new or changed in Dynamics 365 for Operations platform update 5</td>
</tr>
<tr>
<td>Platform update 4</td>
<td>7.0.4425.16161</td>
<td>February 2017</td>
<td>What's new or changed in Dynamics 365 for Operations platform update 4</td>
</tr>
<tr>
<td>Platform update 3</td>
<td>7.0.4307.16141</td>
<td>November 2016</td>
<td>What's new or changed in Dynamics 365 for Operations platform update 3</td>
</tr>
<tr>
<td>Platform update 2</td>
<td>7.0.4230.16130</td>
<td>August 2016</td>
<td>What's new or changed in Dynamics AX platform update 2</td>
</tr>
<tr>
<td>Platform update 1</td>
<td>7.0.4127.16103</td>
<td>May 2016</td>
<td>What's new or changed in Dynamics AX platform update 1</td>
</tr>
<tr>
<td>Dynamics AX 7.0</td>
<td>7.0.4030.16079</td>
<td>February 2016</td>
<td>What's new or changed in Dynamics AX 7.0</td>
</tr>
</tbody>
</table>
This topic lists the features that are included in the platform updates for version 10.0.23 of Finance and Operations apps. This version has a build number of 7.0.6206 and is available on the following schedule:

- **Preview of release**: October 2021
- **General availability of release (self-update)**: December 2021

### Features included in this release

The following table lists the features that are included in this release.

<table>
<thead>
<tr>
<th>FEATURE AREA</th>
<th>FEATURE</th>
<th>MORE INFORMATION</th>
<th>ENABLED BY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client features</td>
<td>Open attachments in a new window</td>
<td>Configure document management</td>
<td>Option on the Document management parameters page</td>
</tr>
<tr>
<td>Client features</td>
<td>Auto resize a grid column to fit contents</td>
<td>Grid capabilities</td>
<td>Included with the New grid control feature, which is enabled by default in Feature management.</td>
</tr>
</tbody>
</table>

**IMPORTANT**

Some or all of the functionality noted in this topic is available as part of a preview release. The content and the functionality are subject to change. For more information about preview releases, see [Service update availability](#).
For information about the bug fixes that are included in this update, sign in to Microsoft Dynamics Lifecycle Services (LCS), and view the KB article.

**Dynamics 365: 2021 release wave 2 plan**

Wondering about upcoming and recently released capabilities in any of our business apps or platform?

Check out the [Dynamics 365: 2021 release wave 2 plan](#). We've captured all the details, end to end, top to bottom, in a single document that you can use for planning.

**Removed and deprecated platform features**

The [Removed or deprecated platform features](#) topic describes features that have been removed, or that are planned for removal in platform updates of Finance and Operations apps.

- A *removed* feature is no longer available in the product.
- A *deprecated* feature isn't in active development and might be removed in a future update.

A deprecation notice will be added in the [Removed or deprecated platform features](#) topic 12 months before the removal of any feature from the product.

For breaking changes that affect only compilation time, but that are binary-compatible with sandbox and production environments, the deprecation time will be less than 12 months. Typically, these changes are functional updates that must be made to the compiler.
This topic lists the features that are included in the platform updates for version 10.0.22 of Finance and Operations apps. This version has a build number of 7.0.6164 and is available on the following schedule:

- **Preview of release**: September 2021
- **General availability of release (self-update)**: October 2021
- **General availability of release (auto-update)**: November 2021

**Features included in this release**

The following features are included in this release. Some of the listed features are still in preview, whereas others might already be generally available. For the official release date of each feature, see the release plan.

<table>
<thead>
<tr>
<th>FEATURE AREA</th>
<th>FEATURE</th>
<th>MORE INFORMATION</th>
<th>ENABLED BY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client features</td>
<td>Open-source software update – upgrade Moment and remove jQWidgets</td>
<td>Not applicable</td>
<td>Feature management</td>
</tr>
<tr>
<td>Client features</td>
<td>New color picker control</td>
<td>Not applicable</td>
<td>Feature management</td>
</tr>
<tr>
<td>Client features</td>
<td><strong>Visual updates to the Hierarchy viewer control</strong></td>
<td>HierarchyViewer control</td>
<td>Default</td>
</tr>
</tbody>
</table>

Modifications were made to the HierarchyViewer control to improve its accessibility, especially for 400-percent zoom scenarios. These modifications included restyling the control so that it’s aligned with the Fluent design language, to help readability of the control at all zoom levels.
Batch OData API

The batch functionality now exposes an Open Data Protocol (OData) application programming interface (API) that can be used to requeue batch jobs. Customers can use the OData endpoint to requeue batch jobs that are in a terminal state. This feature can be integrated with any automation by using Microsoft Power Automate, custom APIs, and so on.

New scenarios enabled in the Microsoft Power Platform integration.

Here are some examples:

- Integration setup
- Automated setup for dual-write and virtual entities
- Streamlined user setup
- Finance and Operations apps business events and data events in Microsoft Dataverse
- Improved development tools
- Enhanced add-in experience

**Additional resources**

**Bug fixes**

For information about the bug fixes that are included in this update, sign in to Microsoft Dynamics Lifecycle Services (LCS), and view the KB article.

**Dynamics 365: 2021 release wave 2 plan**

Wondering about upcoming and recently released capabilities in any of our business apps or platform?

Check out the Dynamics 365: 2021 release wave 2 plan. We've captured all the details, end to end, top to bottom, in a single document that you can use for planning.

**Removed and deprecated platform features**

The Removed or deprecated platform features topic describes features that have been removed, or that are
planned for removal in platform updates of Finance and Operations apps.

- A *removed* feature is no longer available in the product.
- A *deprecated* feature isn’t in active development and might be removed in a future update.

A deprecation notice will be added in the Removed or deprecated platform features topic 12 months before the removal of any feature from the product.

For breaking changes that affect only compilation time, but that are binary-compatible with sandbox and production environments, the deprecation time will be less than 12 months. Typically, these changes are functional updates that must be made to the compiler.
This topic lists the features that are included in the platform updates for version 10.0.21 of Finance and Operations apps. This version has a build number of 7.0.6129 and is available on the following schedule:

- **Preview of release**: August 2021
- **General availability of release (self-update)**: September 2021
- **General availability of release (auto-update)**: October 2021

### Features included in this release

The following features are included in this release. Some of the listed features are still in preview, while others may already be generally available. See the release plan for official release dates for each feature.

Some features must be enabled by using Feature management before you can use them.

<table>
<thead>
<tr>
<th>FEATURE AREA</th>
<th>FEATURE</th>
<th>MORE INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client features</td>
<td>Improved legal entity support for saved views</td>
<td>Saved views</td>
</tr>
<tr>
<td>Client features</td>
<td>Updates to client feature states</td>
<td>Not applicable</td>
</tr>
<tr>
<td>System administration</td>
<td>Enhanced support for full feature lifecycle</td>
<td>Feature management overview</td>
</tr>
<tr>
<td></td>
<td>in Feature management</td>
<td></td>
</tr>
<tr>
<td>Availability monitoring</td>
<td>Synthetic monitoring of Service Fabric environments</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Developer tools</td>
<td>Visual Studio 2019 is now officially supported.</td>
<td>Development tools in Visual Studio</td>
</tr>
</tbody>
</table>

### Features turned on by default in this release

The following table lists the features that are turned on by default in 10.0.21. Most features that have been turned on atomically can be turned off in Feature management.

<table>
<thead>
<tr>
<th>FEATURE NAME</th>
<th>ENABLE DATE</th>
<th>FEATURE ADDED</th>
<th>FEATURE STATE</th>
<th>MODULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable usage of EPPlus library in Electronic reporting framework</td>
<td>9/1/2021</td>
<td>1/6/2020</td>
<td>On by default</td>
<td>Organization administration</td>
</tr>
<tr>
<td>FEATURE NAME</td>
<td>ENABLE DATE</td>
<td>FEATURE ADDED</td>
<td>FEATURE STATE</td>
<td>MODULE</td>
</tr>
<tr>
<td>---</td>
<td>----------------------------</td>
<td>-------------</td>
<td>----------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Convert Electronic Reporting outbound documents from Microsoft Office formats to PDF</td>
<td>9/1/2021</td>
<td>1/6/2020</td>
<td>On by default</td>
<td>Organization administration</td>
</tr>
<tr>
<td>Saved views</td>
<td>9/1/2021</td>
<td>6/9/2019</td>
<td>On by default</td>
<td>System administration</td>
</tr>
<tr>
<td>New grid control</td>
<td>9/1/2021</td>
<td>10/7/2019</td>
<td>On by default</td>
<td>System administration</td>
</tr>
<tr>
<td>Grouping in grids</td>
<td>9/1/2021</td>
<td>12/19/2019</td>
<td>On by default</td>
<td>System administration</td>
</tr>
<tr>
<td>Email distributor batch job email expiration</td>
<td>9/1/2021</td>
<td>2/19/2020</td>
<td>On by default</td>
<td>System administration</td>
</tr>
<tr>
<td>Designate fields as required using personalization</td>
<td>9/1/2021</td>
<td>4/1/2020</td>
<td>On by default</td>
<td>System administration</td>
</tr>
<tr>
<td>Show related document attachments</td>
<td>9/1/2021</td>
<td>4/24/2020</td>
<td>Mandatory</td>
<td>System administration</td>
</tr>
<tr>
<td>Report PDF viewer</td>
<td>9/1/2021</td>
<td>4/24/2020</td>
<td>On by default</td>
<td>System administration</td>
</tr>
<tr>
<td>Enable Export on Report PDF viewer</td>
<td>9/1/2021</td>
<td>6/14/2020</td>
<td>On by default</td>
<td>System administration</td>
</tr>
<tr>
<td>Enable Network Printing on Report PDF viewer</td>
<td>9/1/2021</td>
<td>6/14/2020</td>
<td>On by default</td>
<td>System administration</td>
</tr>
<tr>
<td>Key vault client caching</td>
<td>9/1/2021</td>
<td>8/17/2020</td>
<td>On by default</td>
<td>System administration</td>
</tr>
<tr>
<td>Allow validation of control state in task recordings</td>
<td>9/1/2021</td>
<td>8/17/2020</td>
<td>Mandatory</td>
<td>System administration</td>
</tr>
<tr>
<td>Enable drill through links on Report PDF viewer control.</td>
<td>9/1/2021</td>
<td>8/17/2020</td>
<td>On by default</td>
<td>System administration</td>
</tr>
</tbody>
</table>
### New feature details

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Enable Date</th>
<th>Feature Added</th>
<th>Feature State</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freezing columns in grids</td>
<td>9/1/2021</td>
<td>2/20/2021</td>
<td>On by default</td>
<td>System administration</td>
</tr>
<tr>
<td>New color picker control</td>
<td>9/1/2021</td>
<td>9/1/2021</td>
<td>On by default</td>
<td>System administration</td>
</tr>
<tr>
<td>Prevents multiple executing Workflow message processing batch jobs.</td>
<td>9/1/2021</td>
<td>9/1/2021</td>
<td>On by default</td>
<td>Workflow</td>
</tr>
<tr>
<td>Resets the batch affinity</td>
<td>9/1/2021</td>
<td>9/1/2021</td>
<td>On by default</td>
<td>Workflow</td>
</tr>
<tr>
<td>Workflow notification actions</td>
<td>9/1/2021</td>
<td>8/22/2020</td>
<td>On by default</td>
<td>Workflow</td>
</tr>
</tbody>
</table>

### Additional resources

**Bug fixes**

For information about the bug fixes that are included in this update, sign in to Microsoft Dynamics Lifecycle Services (LCS), and view the KB article.

**Dynamics 365: 2021 release wave 2 plan**

Wondering about upcoming and recently released capabilities in any of our business apps or platform?

Check out the Dynamics 365: 2021 release wave 2 plan. We’ve captured all the details, end to end, top to bottom, in a single document that you can use for planning.

**Removed and deprecated platform features**

The Removed or deprecated platform features topic describes features that have been removed, or that are planned for removal in platform updates of Finance and Operations apps.

- A *removed* feature is no longer available in the product.
- A *deprecated* feature isn’t in active development and might be removed in a future update.

A deprecation notice will be added in the Removed or deprecated platform features topic 12 months before the removal of any feature from the product.

For breaking changes that affect only compilation time, but that are binary-compatible with sandbox and production environments, the deprecation time will be less than 12 months. Typically, these changes are functional updates that must be made to the compiler.
Removed or deprecated features in Finance and Operations apps

The following topics provide information about removed or deprecated features in Finance and Operations apps:

- Removed or deprecated features in Dynamics 365 Commerce
- Removed or deprecated features in Dynamics 365 Finance
- Removed or deprecated features in Dynamics 365 Supply Chain Management
- Removed or deprecated platform features
- Removed or deprecated features in previous releases

Deprecations in the Microsoft Power Platform

The following topic provides information about deprecations in the Microsoft Power Platform:

- Important changes (deprecations) coming in Power Apps, Power Automate, and customer engagement apps
This topic describes features that have been removed, or that are planned for removal in platform updates of Finance and Operations apps.

- A removed feature is no longer available in the product.
- A deprecated feature is not in active development and may be removed in a future update.

This list is intended to help you consider these removals and deprecations for your own planning.

Detailed information about objects in Finance and Operations apps can be found in the Technical reference reports. You can compare the different versions of these reports to learn about objects that have changed or been removed in each version of Finance and Operations apps.

## Feature removal effective October 2021

### Microsoft Azure SQL reports in Lifecycle Services (LCS)

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>All activities and monitoring will be performed internally, by the platform, through automation. This will not require any manual intervention.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, there is now an automated system, which renders these capabilities obsolete.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>SQL reports: Current DTU, Current DTU Details, Get Lock Details, List of Current Plan Guide, Get List of Query ID’s, Get the SQL query plan for a given Plan ID, Get query plans and execution status, Get throttle config, Get wait stats, List most expensive queries</td>
</tr>
<tr>
<td>Deployment option</td>
<td>Cloud deployment: Affects Microsoft-managed production environments and Tier 2 through Tier 5 sandbox environments.</td>
</tr>
<tr>
<td>Status</td>
<td>Removed</td>
</tr>
</tbody>
</table>

### Azure SQL actions in LCS

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>We are deprecating some SQL actions in LCS. All activities and monitoring will be performed internally, by the platform, through automation. This will not require any manual intervention.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, there is now an automated system, which renders these capabilities obsolete.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>SQL actions: Create a plan guide to force Plan ID, Create a plan guide to add table hints, Remove Plan guide, Disable/Enable page locks and lock escalation, Update statistics on a table, Rebuild Index, Create Index</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Deployment option</td>
<td>Cloud deployment: Affects Microsoft-managed production environments and Tier 2 through Tier 5 sandbox environments.</td>
</tr>
<tr>
<td>Status</td>
<td>Removed</td>
</tr>
</tbody>
</table>

**Feature deprecation effective October 2021**

*Show related document attachments* feature

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The feature was returning unexpected results.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No. Any further plans regarding this functionality will be communicated through our standard release wave disclosure process.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Web client - Document attachment experience</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated</td>
</tr>
</tbody>
</table>

**Platform updates for version 10.0.23 of Finance and Operations apps**

OnDBSynchronize event

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>There is no control to execute this event.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, move existing methods subscribed to by the OnDBSynchronize event to a SysSetup extended class.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Database synchronization</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated. Planned removal date is October 2022.</td>
</tr>
</tbody>
</table>

SystemNotificationsManager.AddNotification API

| Reason for deprecation/removal | Microsoft requires additional parameters when adding notifications.                                                                                                                            |


### Replaced by another feature?
Yes, the `SystemNotificationsManager.AddSystemNotification()` API. This API requires that you explicitly set `ExpirationDateTime` and `RuleID` for generated notifications.

### Product areas affected
Web client

### Deployment option
All

### Status
Deprecated. Planned removal date is April 2023.

## Platform updates for version 10.0.21 of Finance and Operations apps

### Skype for Business Online support

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Skype for Business Online has been retired. For more information, see The Skype for Business Online service has retired.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Not currently, although we may consider adding presence from Teams in the future.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Web client</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated. The <strong>Skype enabled</strong> setting has been turned off starting in release 10.0.21. The removal of this setting is targeted for April 2022; however, the feature will stop functioning after the Skype team shuts down the service.</td>
</tr>
</tbody>
</table>

## Feature deprecation effective August 2021

### Microsoft Azure SQL reports in Lifecycle Services (LCS)

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>All activities and monitoring will be performed internally, by the platform, through automation. This will not require any manual intervention.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, there is now an automated system, which renders these capabilities obsolete.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>SQL reports: Current DTU, Current DTU Details, Get Lock Details, List of Current Plan Guide, Get List of Query ID's, Get the SQL query plan for a given Plan ID, Get query plans and execution status, Get throttle config, Get wait stats, List most expensive queries</td>
</tr>
</tbody>
</table>
### Azure SQL actions in LCS

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>We are deprecating some SQL actions in LCS. All activities and monitoring will be performed internally, by the platform, through automation. This will not require any manual intervention.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, there is now an automated system, which renders these capabilities obsolete.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>SQL actions: Create a plan guide to force Plan ID, Create a plan guide to add table hints, Remove Plan guide, Disable/Enable page locks and lock escalation, Update statistics on a table, Rebuild Index, Create Index</td>
</tr>
<tr>
<td>Deployment option</td>
<td>Cloud deployment: Affects Microsoft-managed production environments and Tier 2 through Tier 5 sandbox environments.</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: Planned removal date is October 2021.</td>
</tr>
</tbody>
</table>

### Feature deprecation effective May 2021

**Globalization portal in Lifecycle Services (LCS)**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>We are deprecating the Globalization portal in LCS as this feature has been superseded by other LCS-based services.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, this feature is replaced by LCS issue search and Dynamics regulatory alert submission service.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Globalization portal in LCS</td>
</tr>
<tr>
<td>Deployment option</td>
<td>Cloud deployment</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: Planned removal date is May 2022.</td>
</tr>
</tbody>
</table>

### Feature removed effective January 28, 2021

**Batch job to handle SQL index defragmentation**
<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>In order to reduce the overhead of operating, monitoring, and maintaining the index management by customers, this feature has been removed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Going forward, the index maintenance will be performed by Microsoft services. This will happen continuously without affecting the user workloads.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Finance and Operations apps</td>
</tr>
<tr>
<td>Deployment option</td>
<td>Cloud deployment - affects Microsoft-managed production environments and Tier 2 through Tier 5 sandbox environments.</td>
</tr>
<tr>
<td>Status</td>
<td>This feature is removed.</td>
</tr>
</tbody>
</table>

### Platform updates for version 10.0.17 of Finance and Operations apps

#### Visual Studio 2015

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>To support the latest versions of Visual Studio, some changes have to be made to the X++ extensions for Visual Studio. These changes are incompatible with Visual Studio 2015.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Visual Studio 2017 will replace Visual Studio 2015 as the deployed and required version.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Visual Studio development tools</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: Upon updating, the previous X++ tools will be removed from Visual Studio 2015, and the updated tools will not install on Visual Studio 2015. There is no impact on hosted builds. For build virtual machines, the build pipeline (build definition) needs to be manually updated to change the dependency from MSBuild 14.0 (Visual Studio 2015) to MSBuild 15.0 (Visual Studio 2017) as described in <a href="#">Update a legacy pipeline in Azure Pipelines</a>.</td>
</tr>
</tbody>
</table>

#### User avatar

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The user avatar that displays on the right side of the navigation bar was retrieved using an API from the Dynamics 365 header control, which has been deprecated.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Users will see their initials in a circle in the navigation bar instead. This is the same visual currently used on development machines.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product areas affected</th>
<th>Web client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of version 10.0.17</td>
</tr>
</tbody>
</table>

**Enterprise Portal (EP) deprecation**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The metadata artifacts associated with Dynamics AX 2012 Enterprise Portal (EP) have been deprecated, as EP was never supported in the Finance and Operations apps.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Web client</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: All EP code is scheduled to be removed in the October 2021 release.</td>
</tr>
</tbody>
</table>

**Platform updates for version 10.0.15 of Finance and Operations apps**

**Internet Explorer 11 support for Dynamics 365 is deprecated**

| Reason for deprecation/removal | Effective December 2020, Microsoft Internet Explorer 11 support for all Dynamics 365 products is deprecated, and Internet Explorer 11 won’t be supported after August 2021.  
This will impact customers who use Dynamics 365 products that are designed to be used through an Internet Explorer 11 interface. After August 2021, Internet Explorer 11 won’t be supported for such Dynamics 365 products. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>We recommend that customers transition to Microsoft Edge.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>All Dynamics 365 products</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: Internet Explorer 11 won’t be supported after August 2021.</td>
</tr>
</tbody>
</table>

**Visual Studio add-in to apply metadata hotfixes**


### Metadata hotfixes

**Reason for deprecation/removal**

Metadata hotfixes are no longer supported with the One Version service updates that were introduced in July 2018 with version 8.1.

**Replaced by another feature?**

Individual metadata hotfixes are not available for supported versions. Cumulative quality updates are applied instead.

**Product areas affected**

Visual Studio add-ins

**Deployment option**

Development virtual machines

**Status**

With version 10.0.15, the add-in is no longer included in the Visual Studio tools.

### Platform updates for version 10.0.14 of Finance and Operations apps

#### Online users page

**Reason for deprecation/removal**

This is a legacy page that was built for previous client/server architecture. The information on this page is not always accurate, which can be confusing and misleading.

**Replaced by another feature?**

We will provide a new page in a future update.

**Product areas affected**

System Administration

**Deployment option**

All

**Status**

By October 2021 this form will be removed.

### Platform updates for version 10.0.13 of Finance and Operations apps

#### Custom code defined in SSRS report properties

**Reason for deprecation/removal**

In general, custom code offers limited benefits while at the same time, requires significant resourcing and compute to support. Custom code is primarily used by report authors to call public methods from a custom code assembly. However, the cloud-hosted service does not support references to custom assemblies for SSRS reports.

**Replaced by another feature?**

Report authors may choose to continue referencing public .NET APIs for Math, Conversion, and Format operations from any textbox expression. For more information, see Add Code to a Report (SSRS).

**Product areas affected**

Subset of application report designs defined in RDL that contain custom code.
With version 10.0.13, the compiler will begin issuing a warning for instances where custom code is detected in an SSRS report definition. To fix the issue, open the report design definition and remove all custom code artifacts. This warning will be replaced with a compiler error in a future update.

Upgrade of three jQuery component libraries

Reason for deprecation/removal
Three jQuery component libraries are being updated for security fixes and to maintain currency.

Replaced by another feature?
The following libraries are being affected: jQuery (to version 3.5.0 from version 2.1.4), jQuery UI (to version 1.12.1 from version 1.11.4), jQuery qTip (to version 3.0.3 from 2.2.1). Migration guidance has been provided online by jQuery.

Product areas affected
Extensible controls, specifically custom JavaScript code utilizing deprecated or removed APIs

Deployment option
All

Status
With version 10.0.13/Platform update 37, customers can optionally move to the latest libraries by enabling the “Upgrade three jQuery component libraries” feature. Moving to the new libraries will be mandatory with the April 2021 release to allow time for migration of affected APIs.

Existing grid control/forceLegacyGrid() API

Reason for deprecation/removal
The existing grid control is being replaced by the new grid control.

Replaced by another feature?
The new grid control

Product areas affected
Web client

Deployment option
All

Status
In version 10.0.13, the new grid control is generally available, and customers can optionally turn on this feature. The new grid control will become on by default with the October 2021 release and is currently targeted to be mandatory in April 2022. When the new grid control becomes mandatory, the forceLegacyGrid() API will no longer be honored.

Personalization without saved views
### Platform updates for version 10.0.13 of Finance and Operations apps

**Reason for deprecation/removal**
The saved views feature has been introduced to allow for better performance and additional capabilities.

**Replaced by another feature?**
Saved views

**Product areas affected**
Web client

**Deployment option**
All

**Status**
In version 10.0.13/Platform update 37, the saved views feature is generally available, and customers can optionally turn on this feature. The saved views feature will become mandatory in the October 2021 release.

### Platform updates for version 10.0.12 of Finance and Operations apps

**Grid or group control form extensions containing invalid field references**

**Reason for deprecation/removal**
The data group property on grid or group controls is used to automatically show all the fields of a field group. A grid or group control added by extension could contain fields that are no longer defined on the field group, or it might be missing fields that are defined on the field group. This can cause inconsistent behavior at runtime. Platform updates for version 10.0.12 of Finance and Operations apps now categorize this issue as a compiler warning. To fix this issue, open the form extension and save it.

**Replaced by another feature?**
This compiler warning will be replaced with a compiler error in a future update.

**Product areas affected**
Visual Studio development tools

**Deployment option**
All

**Status**
A compiler warning is introduced in platform updates for version 10.0.12 of Finance and Operations apps.

### Platform updates for version 10.0.11 of Finance and Operations apps

**Explicit safe lists for self-service environments**

**Reason for deprecation/removal**
The process for moving IP to safe lists has changed. Self-service no longer supports IP safe lists.

**Replaced by another feature?**
For more information, see Configuring Azure Active Directory Conditional Access.
<table>
<thead>
<tr>
<th>Product areas affected</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment option</td>
<td>Cloud</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: This feature is fully-deprecated for self-service deployments.</td>
</tr>
</tbody>
</table>

**Visual Studio 2015**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>To support the latest versions of Visual Studio, some changes have to be made to the X++ extensions for Visual Studio. These changes are incompatible with Visual Studio 2015.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Visual Studio 2017 will replace Visual Studio 2015 as the deployed and required version.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Visual Studio development tools</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Virtual machines deployed on version 10.0.13 (Platform update 37) or later contain Visual Studio 2017. Version 10.0.16 (Platform update 40) is the final release with support for Visual Studio 2015. Virtual machines with only Visual Studio 2015 will not be able to update to version 10.0.17 (Platform update 41).</td>
</tr>
</tbody>
</table>

**Field groups containing invalid field references**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Field groups in table metadata definitions can contain field references that aren't valid. If these field groups are deployed, they can cause runtime failures in Financial Reporting and Microsoft SQL Server Reporting Services (SSRS). Platform update 23 introduced a compiler warning that enabled this metadata issue to be addressed. Platform updates for version 10.0.11 of Finance and Operations apps categorize this issue as a compiler error. To fix this issue, follow these steps.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>This compiler error permanently replaces the compiler warning.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Visual Studio development tools</td>
</tr>
</tbody>
</table>
### ISV licenses created by using the SHA1 hashing algorithm

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The process for creating independent software vendor (ISV) licenses has changed. For more information, see Independent software vendor (ISV) licensing.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes. Use Windows PowerShell to create licenses.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Visual Studio development tools</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: ISV licenses that were created by using the SHA1 hashing algorithm. This algorithm depended on certificates that were created by using the MakeCert utility, and this utility has been deprecated. Deprecated: The use of SHA1 for security or hashing purposes. SHA1 will cease to function in early 2021. Therefore, it should no longer be used. Removed: Support for Transport Layer Security (TLS) 1.0 and TLS 1.1 incoming or outgoing requests.</td>
</tr>
</tbody>
</table>

### Platform update 32

#### Workflow request change dialog box no longer includes user selection drop-down list

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>This was a security issue because the request for change could be sent to an unintended user. This was also a usability issue because it forced the user to determine who the workflow originator was and manually select them.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Workflow</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>The user selection drop-down list was removed from the request change dialog box in Platform update 32. Request change requests will be automatically sent to the originator as intended. For more information about this functionality, see Actions in workflow approval processes.</td>
</tr>
</tbody>
</table>
Embedded drill-through links are no longer supported in paginated documents rendered by the cloud-hosted service

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Navigation URLs embedded in documents rendered by the service may contain sensitive business data. We are removing support for embedded drill-through links in documents as a security precaution to further protect customer's data. Users will also benefit from improved performance while interactively producing documents as a result of this change.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Reporting</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>This feature is actively being removed from the service. The modern client offers numerous options for producing views that include auto-generated links to assist in navigating the application. Paginated documents rendered by the service are recommended for external communications that are emailed, archived, and printed for recipients. We have improved the experience for previewing documents directly in the browser, which offers direct access to local printers. For more information, see Preview PDF documents with an embedded viewer.</td>
</tr>
</tbody>
</table>

Previous announcements about removed or deprecated features

To learn more about features that have been removed or deprecated in previous releases, see Removed or deprecated features in previous releases.
This topic describes features that have been removed or deprecated from Dynamics 365 for Finance and Operations and previous releases of that product.

- A **removed** feature is no longer available in the product.
- A **deprecated** feature is not in active development and may be removed in a future update.

This list is intended to help you consider these removals and deprecations for your own planning.

Detailed information about objects in Finance and Operations apps can be found in the Technical reference reports. You can compare the different versions of these reports to learn about objects that have changed or been removed in each version of Finance and Operations apps.

### Finance 10.0.7 with Platform update 31

**Chinese voucher types without Account groups selection**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Changed to the feature with account groups selection.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Application</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: By December 1, 2020, we plan to no longer support Chinese voucher types setup without Account groups selection. Find more details about new feature design in What’s new in 10.0.7</td>
</tr>
</tbody>
</table>

### Finance and Operations 10.0.6 with Platform update 30

**DimensionHash.getHash(str _message)**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Windows is deprecating the use of SHA1, as documented in Windows Enforcement of SHA1 Certificates.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature</td>
<td>Replaced by another feature?</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>Hash.ComputeSHA1Hash(string message)</td>
<td>Yes</td>
</tr>
<tr>
<td>FormDateTimeControl.setUtcString()</td>
<td>Yes</td>
</tr>
<tr>
<td>Blocklist report (IT) – Feature reference IT-00001</td>
<td>No</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: By October 1, 2020, we plan to no longer support this report.</td>
</tr>
</tbody>
</table>

**Domestic tax report – Feature reference IT-00003**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Not legally required.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Italian localization</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: By October 1, 2020, we plan to no longer support the Domestic tax report – Feature reference IT-00003.</td>
</tr>
</tbody>
</table>

**October 2019 deprecation announcement**

**Flowchart diagrams in Business process modeler**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>We are deprecating the flowchart diagrams component in Business process modeler (BPM), because the legacy design caused low usage.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Areas affected</td>
<td>Business process modeler</td>
</tr>
</tbody>
</table>
| Status                        | Deprecated: The flowchart diagrams component in BPM is expected to be removed in 2020. The following functionality will be unavailable:  
  - All flowcharts will be read-only and unavailable for editing. The shape properties that are associated with flowchart activities will also be unavailable. These flowcharts include both the default flowcharts that are automatically generated and customized flowcharts that are modified based on those default flowcharts.  
  - The process steps will be read-only and unavailable for editing.  
  - The legacy fit/gap analysis feature will be unavailable. Therefore, no gap list will be automatically created or available for export.  
  Note: This feature had previously been deprecated and replaced by Microsoft Azure DevOps integrations.  
  - The version history of the flowchart will be unavailable. |
### US Payroll tax updates

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>We are retiring tax updates for the US Payroll functionality due to low usage and enhanced functionality that is now offered via strategic integrations.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Payroll</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: By July 31, 2024, we plan to no longer provide tax updates to US Payroll customers. The functionality will remain in the product, but enhancements will no longer keep the functionality up to date, and any product defects will be evaluated on a case-by-case basis.</td>
</tr>
</tbody>
</table>

**NOTE**

This represents a change from the original discontinuation date of October 1, 2021. For more information, see [Tax updates being retired for US Payroll feature in Microsoft Dynamics 365 for Finance and Operations](#).

### Data management staging clean up

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Does not meet the core requirements that are needed for scheduling periodic cleanup.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, the Job history cleanup feature is being added to meet the scenarios holistically.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Data management</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: Target timeframe for the functionality to be removed is December 2020.</td>
</tr>
</tbody>
</table>

### Finance and Operations 10.0.4 with Platform update 28

**France: FEC Accounting data export in XML**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Replaced by TXT format, French FEC audit file is available through General ledger &gt; Periodic tasks &gt; Data export.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
<td>General ledger</td>
</tr>
<tr>
<td><strong>Deployment option</strong></td>
<td>All</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Deprecated. Target timeframe for the functionality to be removed is July 2020.</td>
</tr>
</tbody>
</table>

**Legacy navigation bar**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Header alignment with other Dynamics and Office products. For more details, see Updated navigation bar that aligns with the Office header.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>Starting in Platform update 24, a restyled navigation bar that features search was introduced.</td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
<td>Web client</td>
</tr>
<tr>
<td><strong>Deployment option</strong></td>
<td>All</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Deprecated: Starting in April 2020, the legacy navigation bar will no longer be available. Until that point, customers can revert to the legacy navigation bar through the Client performance options page.</td>
</tr>
</tbody>
</table>

**Finance and Operations 10.0.2 with Platform update 26**

**Legacy default action behavior**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The legacy behavior for default actions in grids results in an unexpected column having the default action link after grid columns have been reordered via personalization. The new sticky default action feature corrects this. For more details, see Sticky default actions in grids.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>Starting in Platform update 21, a feature for “sticky default actions” was introduced. This feature can be enabled on the Client performance options page.</td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
<td>Grids in the web client</td>
</tr>
<tr>
<td><strong>Deployment option</strong></td>
<td>All</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Deprecated: Starting in April 2020, sticky default actions will be the default behavior, without a mechanism to revert to the legacy behavior.</td>
</tr>
</tbody>
</table>
### Legacy "is one of" filtering experience

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The &quot;is one of&quot; filtering experience went through a redesign in Platform update 22, with the plan for this to eventually be the only &quot;is one of&quot; filtering experience.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Starting in Platform update 22, an improved &quot;is one of&quot; filtering experience became available on the Client performance options page. For more information, see <a href="#">Optimized is one of filtering experience</a>.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Web client</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: Starting in April 2020, the improved &quot;is one of&quot; experience will be the default behavior, without a mechanism to revert to the legacy behavior.</td>
</tr>
</tbody>
</table>

### Parameter to enable sales orders with multiple project contract funding sources

Support for creating project-based sales orders where the project contract has multiple funding sources is enabled with the Project management parameters setting [Allow sales orders for project with multiple funding sources](#). By default, this parameter is not enabled.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The functionality will always be enabled after the parameter is removed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No. The functionality to support project-based sales orders with multiple funding sources will always be enabled.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>The <a href="#">Allow sales orders for projects with multiple funding sources</a> parameter will be removed. The following methods will be modified when the parameter is removed:&lt;&gt;ctrlSalesOrderTable method in ProjStatusType class, validate method for ProjId field, and run method in SalescreateOrder form. The following methods will be deprecated when the parameter is removed: IsSalesOrderAllowedForMultipleFundingSources in ProjTable table file, IsAllowSalesOrdersForMultipleFundingSourcesParam Enabled method in ProjTable table file, AllowSalesOrdersForMultipleFundingSources data field in ProjParameters form and ProjParameterEntity files, IsAssociatedToMultipleFundingSourcesContract private method in ProjTable table file.</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecation is planned for the April 2020 release wave.</td>
</tr>
</tbody>
</table>

### Legacy workflow reports for tracking and instance status

- Support for creating project-based sales orders where the project contract has multiple funding sources is enabled with the [Project management parameters](#) setting [Allow sales orders for project with multiple funding sources](#). By default, this parameter is not enabled.
<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The legacy workflow reports for tracking and instance status are being deprecated because they are no longer referenced from the navigation. The report names are WorkflowWorkflowInstanceByStatusReport and WorkflowWorkflowTrackingReport.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>The workflow history form can be used instead.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Web client</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: Target timeframe for the functionality to be removed is April 2020.</td>
</tr>
</tbody>
</table>

**Finance and Operations 10.0.1 with Platform update 25**

**Deprecated APIs and potential breaking changes**

**Deriving from internal classes is deprecated**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Before Platform update 25, it was possible to create a class or table that derives from an internal class/table that is defined in another package/module. This is not a safe coding practice. As of Platform update 25, the compiler will display a warning.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>The compiler warning will be replaced by an error in Platform update 26. This change is backward compatible at runtime, which means that Platform update 25 or newer can be deployed on any sandbox or production environment without the need to modify custom code. This change only affects development and compile time.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Visual Studio development tools</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: The warning will become a compilation error in Platform update 26.</td>
</tr>
</tbody>
</table>

**Overriding internal methods is deprecated**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Before Platform update 25, it was possible to override an internal method in a derived class that is defined in another package/module. This is not a safe coding practice. As of Platform update 25, the compiler will display a warning.</th>
</tr>
</thead>
</table>
This warning will be replaced by a compile error in Platform update 26. This change is backward compatible at runtime, which means that Platform update 25 or newer can be deployed on any sandbox or production environment without the need to modify custom code. This change only affects development and compile time.

Visual Studio development tools

All

Deprecated: The warning will become a compilation error in Platform update 26.

Visual Studio development tools

All

Removed as of Finance and Operations 10.0.0 with Platform update 24.

Product information management

All

Removed as of Finance and Operations 8.1.3 with Platform update 23.

Customers can use the Export action provided by the embedded SQL Server Reporting Services (SSRS) ReportViewer control to download documents produced by Finance and Operations applications. This HTML-based presentation of the report offers users a non-paginated preview of the document.

The non-paginated nature of the HTML-based preview experience does not deliver fidelity with the physical documents ultimately produced by Finance and Operations. By fully embracing PDF as the standard format for business documents, users are able to take advantage of a modern viewing experience with improved performance when producing application reports.
### Replaced by another feature?

Going forward, PDF documents will be the default format for reports rendered by Finance and Operations.

### Product areas affected

This change does **not** impact customer scenarios where reports are distributed electronically or sent directly to printers.

### Deployment option

All

### Status

Deprecated: A removal date has not been set for this feature. The functionality to automatically preview application reports using an embedded PDF viewer is planned for the May 2019 Platform update.

### Client KPI controls

Embedded key performance indicators (KPIs) could be modeled in Visual Studio by a developer and further customized by the end user.

### Reason for deprecation/removal

The native client controls used to define KPIs have low customer uptake and rely on a developer to add trackable metrics.

### Replaced by another feature?

PowerBI.com service delivers world-class tooling for defining and managing KPIs based on data from external sources. In an upcoming release, we plan to enable you to embed solutions hosted on PowerBI.com in application workspaces.

### Product areas affected

This update will prevent developers from introducing new KPI controls in Visual Studio designer.

### Deployment option

All

### Status

Deprecated: A removal date has not been set for this feature.

### Deprecated APIs and future breaking changes

**Field groups containing invalid field references**

- field references
- invalid
- field
- references
Reason for deprecation/removal

It is possible for table metadata definitions to have field groups containing invalid field references. If deployed, this can cause runtime failures in Financial Reporting and SQL Server Reporting Services (SSRS). This issue is currently categorized as a compiler warning rather than an error, meaning that the deployable package creation and deployment can proceed without fixing the issue. To fix this issue:

1. Remove the invalid field reference from the table field group definition.
2. Recompile.
3. Ensure any warnings or errors are addressed.

Replaced by another feature?

This warning will be replaced by a compile error in the future.

Product areas affected

Visual Studio development tools

Deployment option

All

Status

Deprecated: The warning is a compile-time error with platform updates for version 10.0.11 of Finance and Operations apps.

Complete list

To access the full list of APIs that are being deprecated, see Deprecation of methods and metadata elements.

Finance and Operations 8.1 with Platform update 20

Batch transfer rules for subledger journal account entries

The Synchronous transfer mode is being deprecated in the General ledger parameters. This mode is replaced by Asynchronous and scheduled batch only, which already exist as options for transfer. For additional information, see the General Ledger Parameters – Batch transfer rules blog.

Reason for deprecation/removal

We are removing the synchronous option due to performance impact to the system.

Replaced by another feature?

Asynchronous and scheduled batch are options to use in place of Synchronous.

Product areas affected

General Ledger, Accounts payable, Accounts Receivable, Procurement, Expense

Deployment option

All

Status

Deprecated: Target timeframe for the functionality to be removed is the 10.0 version.

Electronic reporting for Russia

Feature for configuring .txt and .xml file formats of declarations.
<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Replaced with Electronic reporting.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>General Ledger</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
</tbody>
</table>

**Financial reports generator for Russia**  
A tool for setting up data collection for accounting and tax reports, and to export data to XLS and DOC report templates. Functional parts: Export data to XLS and DOC report templates, queries, fixed requisites are removed.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Removed parts are replaced with Electronic reporting.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes. Financial reports setup user interface should be used for setting up data collection rules by GL accounts or tax registers. Export data to various file types, fixed requisites and query-like data collection rules should be configured in Electronic reporting.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>General ledger.</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
</tbody>
</table>

**Integration with external providers for sending electronic reporting through communication channels for Russia**  
Feature exporting generated electronic files of declarations to folder for further sending to official providers of electronic reporting as well as importing state back.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Replaced with electronic messages configurable feature.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>General Ledger, Tax</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
</tbody>
</table>

**Profit tax register wizard**
Feature for creating templates for new profit tax registers. This feature creates X++ objects for new registers, which are then created as templates with the appropriate calculation logic added in.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Feature is not compatible with the Finance and Operations extensibility model.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Tax</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
</tbody>
</table>

**Payroll and Human Resources for Russia**

Russian country specific module for managing staff administration information, timesheet details for employees, payroll accounting, and creating pay statements.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Payroll is not included in the global strategic focus of the Dynamics 365 portfolio. Partners and ISVs are best positioned to provide payroll functionality that is compliant with local regulations and tax updates.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Russian Payroll and Human Resources Management</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: Target timeframe for the functionality to be removed is one of future updates of the 10.0 version.</td>
</tr>
</tbody>
</table>

**Finance and Operations 8.0 with Platform update 15**

No features have been removed or deprecated with this release. Platform update 15 is cumulative and contains new or changed features from Platform update 13, Platform update 14, and Platform update 15.

**Finance and Operations, Enterprise edition 7.3 with Platform update 12**

**Personalized product recommendations**

Starting February 15, 2018, retailers will no longer be able to display personalized product recommendations on a point of sale (POS) device. For more information, see [Product recommendations overview](#).
We are removing the current version of the product recommendation service as we redesign this feature with a better algorithm and newer retail-oriented capabilities.

No. However, after Spring 2018, we plan to bring back this feature to leverage a new recommendation service.

Personalized product recommendations in POS.

All

Removed as of February 15, 2018. This affects customers running Dynamics 365 for Operations 1611 and later.

The possibility to introduce custom functions to be used in the ER expression builder (for more information, see [Extend the list of Electronic reporting (ER) functions](#)) is not supported any more. Due to changes of the ER APIs, the API to call built-in functions from the ER expression builder became internal and can’t be extended any longer.

None. Whenever a new built-in function is needed, a new extension request must be addressed to the ER framework team.

As a temporary work around while the requested function is under development by the ER team, the required logic can be programmed as a method of a custom application class. This method can be accessed in an ER expression as a property of the added ER data source of the `Application\Class` type that refers to that custom application class.

Electronic reporting framework

All

Removed as of Finance and Operations, Enterprise edition 7.3.

These two reports are no longer supported in Finance and Operations. Instead, the `Inventory aging` report can be used to improve the user experience.

Duplicate functionality

Yes. The two reports have been replaced by the `Inventory aging` report.
<table>
<thead>
<tr>
<th>Product areas affected</th>
<th>Inventory management, Cost management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: The menu items for the two reports have been removed in version 7.3. However, the code for the reports remains in the product. The plan is to remove the code in a future release.</td>
</tr>
</tbody>
</table>

**Power BI content packs available on AppSource**

The **Cost management**, **Financial performance**, and **Retail channel performance** content packs, available on the Microsoft AppSource site, are deprecated as a consequence of product updates in Microsoft Power BI. System administration forms used to deploy these content packs to PowerBI.com are also being deprecated in Finance and Operations.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Product updates in Microsoft Power BI.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>The <strong>Cost management</strong>, <strong>Financial performance</strong>, and <strong>Retail channel performance</strong> content packs, available on the AppSource site, are being replaced by analytical applications which allow for solution integrations at the database level. For more information about analytical applications, see Embedded Power BI in workspaces.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product areas affected</th>
<th>Cost management, Finance, and Retail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment option</td>
<td>Cloud only (Integration with PowerBI.com is not supported in on-premises deployments.)</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: Target timeframe for the functionality removal is Q2 2018.</td>
</tr>
</tbody>
</table>

**Standard UI in data management workspace**

The standard UI in data management is the legacy UI, which is the default UI presented to the users when they visit the data management workspace.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>We are investing in providing new user experiences in the new UI.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>The new UI called Enhanced views is replacing the old UI.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Data management workspace</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: Target timeframe for the functionality to be removed is Q2 2018.</td>
</tr>
</tbody>
</table>
**Excise, Sales Tax, Service Tax for India**
These taxes have been subsumed into Indian GST.

<table>
<thead>
<tr>
<th>Reason for removal or deprecation</th>
<th>These taxes have been subsumed into Indian GST.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Indian GST</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Tax</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**File Validation Utility (FVU) for India**

<table>
<thead>
<tr>
<th>Reason for removal or deprecation</th>
<th>Lack of customer usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Indian withholding tax</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**TDS/TCS certificate for India**
Users can download this from the government portal.

<table>
<thead>
<tr>
<th>Reason for removal or deprecation</th>
<th>Lack of customer usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Indian withholding tax</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Export/import (EXIM) incentive scheme for India**

<table>
<thead>
<tr>
<th>Reason for removal or deprecation</th>
<th>Lack of customer usage</th>
</tr>
</thead>
</table>
Dynamics 365 for Retail 7.2

Personalized product recommendations

Starting February 15, 2018, retailers will no longer be able to display personalized product recommendations on a point of sale (POS) device. For more information, see Product recommendations overview.

Reason for deprecation/removal
We are removing the current version of the product recommendation service as we redesign this feature with a better algorithm and newer retail-oriented capabilities.

Replaced by another feature?
No. However, after Spring 2018, we plan to bring back this feature to leverage a new recommendation service.

Product areas affected
Personalized product recommendations in POS.

Deployment option
All

Status
Removed as of February 15, 2018. This affects customers running Dynamics 365 for Retail 7.2 and later.


Currency conversion for accounting and reporting currencies

Currency conversion for accounting and reporting currencies was introduced when the euro was introduced.

Reason for deprecation/removal
Limited usage and addition of the Copy legal entity functionality as a replacement.

Replaced by another feature?
No, but the Copy legal entity and Configurations features were added to make it easier to move to a company that has changing core requirements.

Product areas affected
Financial management

Status
Deprecated: A removal date has not been set for this feature.
### Warehouse mobile devices portal

Warehouse mobile devices portal (WMDP) was a standalone component that was intended for on-premises self-deployment. This component is no longer supported in Finance and Operations. A native app that improves the user experience has replaced the functionality of WMDP.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Duplicate functionality.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes. This feature has been replaced by Finance and Operations - Warehousing. For more information about setup and prerequisites, see Install and configure the Warehousing app overview.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Warehouse management, Transportation management</td>
</tr>
<tr>
<td>Deployment option</td>
<td>Warehouse mobile devices portal (WMDP) was a standalone component that was intended for on-premises self-deployment.</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: Target timeframe for the functionality to be removed is Q4 2019.</td>
</tr>
</tbody>
</table>

### Advanced bank reconciliation matching rule for manual matching

A matching rule was used to select and mark a bank document when documents were manually matched in the reconciliation worksheet.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Limited usage.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No. Column filtering capabilities should be used to find documents for reconciliation.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Cash and bank management</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of July 2017.</td>
</tr>
</tbody>
</table>

### Dynamics 365 for Operations 1611 with Platform update 3

#### AEB payment formats for Spain

The Consejo Superior Bancario payment formats were used to send remittance files to the bank for customer payments and vendor payments. The content of these formats was determined by the Asociación Española de Banca. It covers Cuaderno 19, 32, 58, 34.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment formats are no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, ISO20022 Credit transfer and Direct debit payment formats for Spain</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable, Accounts receivable</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature</td>
</tr>
</tbody>
</table>

**Bank payments transfer for Lithuania**

Bank payment transfers were generated and printed by using the Payment transfer (LT) export format for Lithuania. The Lithuanian market began to use LITAS, the unified electronic banking system, in 2005.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment formats are no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, ISO20022 Credit transfer payment format for Lithuania</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature</td>
</tr>
</tbody>
</table>

**BBS Direkte Remittering payment formats for Norway**

BBS Direkte Remittering payment formats include customer payment collection export (direct debit) and return message import.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment formats are no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>The AvtaleGiro customer payment format for Norway can be used to generate direct debit messages. Return message import will be implemented in future releases</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable, Accounts receivable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature</td>
</tr>
</tbody>
</table>

**Chart of Accounts tool for Spain**

This tool is used when a chart of accounts in Spain requires major changes. Users can import a new chart of accounts in Microsoft Excel or text format, and can also import financial statements.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Limited usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>General ledger</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature</td>
</tr>
</tbody>
</table>
### Dom80 payment format for Belgium
Legacy Belgian payment format for payment collection (direct debit).

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment format is no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, ISO 20022 Direct debit payment format for Belgium</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts receivable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

### DTA/EZAG payment formats for Switzerland
DTA/EZAG formats are integrated into the ESR system, because they can carry on the reference number. Because the reference number isn’t mandatory, these formats can be used to process any vendor payments. These formats are used by companies that have a bank account in a location other than "Postfinance."

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment formats are no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, ISO20022 Credit transfer payment format for Switzerland</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

### EDIFACT-DIRDEB payment format for Austria
EDIFACT-DIRDEB payment format for payment collection (direct debit).

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment format is no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, ISO 20022 Direct debit payment format for Austria</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts receivable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

### EDIVAT for Belgium
EDIVAT is an obsolete Belgian standard for electronic declaration via secure mail. Dynamics AX 2012 retains the read-only solution to enable access to the historical data.

| Reason for deprecation/removal | The functionality is no longer used. |
### eGiro EDIFACT CREMUL payment import format for Norway

eGiro is based on the international UN EDIFACT CREMUL (Multiple Credit Advice Message) standard that is used for automatic posting of customer payments. In Dynamics AX, eGiro is implemented as a customer payment import format.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment format is no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, the ISO20022 Camt.054 notification import.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts receivable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

### External inventory for Poland

Evidence of goods that are taken from a vendor for sales without purchase. Goods that are handled in external inventory don’t affect standard inventory, and can be sold and then purchased automatically. This process creates real inventory movements.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Replaced by another feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, the core Inbound consignment functionality</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable, Inventory management</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

### Financial reports generator for Eastern Europe

A tool is used to set up data collection for accounting and tax reports, and to export data to XLS and DOC report templates.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Limited usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No. The tool will be replaced by Electronic reporting configurations in future releases.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>General Ledger</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Import of customer payment transactions for Finland**

You can select an import format for Finnish payments to import customer payment transactions from an external file that the bank provides.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment format is no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, the ISO20022 Camt.054 notification import.</td>
</tr>
</tbody>
</table>

**Import of payment transactions into a general ledger journal for Finland**

A format that is specific to Finland is used to import accounting transactions into the general ledger.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment format is no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, the ISO20022 Camt.053 bank statement import using Advanced Bank Reconciliation.</td>
</tr>
</tbody>
</table>

**Integration with Isabel synchronized (CIS) for Belgium**

Isabel is the framework for electronic banking in Europe and is a de-facto standard in Belgium.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Integration with Isabel client has been discontinued.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No. The payment formats that are no longer used are replaced by ISO20022 Credit transfer payment format for Belgium.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product areas affected</th>
<th>Accounts payable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>
**Modifications in the chart of accounts and accounting rules for Spain**

This feature is used for changes in the chart of accounts and accounting rules in Spain. It maps accounts to help transform the old chart of accounts into the new chart of accounts, and compares the previous fiscal year with the new fiscal year, even if they were posted to different account numbers.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Limited usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>General ledger</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Pagamento Fornittori vendor payment format**

Legacy Italian payment format for credit transfers.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment format is no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, ISO20022 Credit transfer payment format for Italy</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Payment export formats for Estonia**

The Telehansa and Teleservice formats are used for bank payment export.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment formats are no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, ISO20022 Credit transfer payment format for Estonia</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Payment file archive for Norway**

When payment files are generated, the file archive automatically archives all files that are created, even files that were previously written or read.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Replaced by another feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature</td>
<td>Details</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>Yes, Electronic reporting archived jobs</td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
<td>Accounts payable, Accounts receivable, Organization administration</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Payment import formats for Estonia**

The Telehansa and TeleTeenus formats are used for bank payment import.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment formats are no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>Yes, the ISO20022 Camt.054 bank notification import.</td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
<td>Accounts receivable</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Payroll information in Human Resources**

Human Resources Payroll information

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>This functionality has been replaced by core Payroll and Human Resources pages.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>Benefits, Earnings, and other related pages that were previously in US Payroll have been reconfigured, and are now part of the core Human Resources configuration to help support external payroll processing. This functionality is accessed by using the Human Resources 1 &gt; Payroll configuration key.</td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
<td>Human Resources, Payroll</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Removed as of Dynamics 365 for Operations version 1611.</td>
</tr>
</tbody>
</table>

**Performance management goal workflow**

Performance management includes goal management and integration with performance reviews.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Performance management was redesigned, and the number of goal pages was reduced to simplify the process.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>No. Goals are visible to managers through the Manager Self Service portal, and can be changed and viewed by the manager.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Human capital management</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics 365 for Operations version 1611.</td>
</tr>
</tbody>
</table>

**Postgirot and Postgirot Utland payment formats for Sweden**

Postgirot and Postgirot Utland payment formats for Sweden.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment formats are no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, ISO20022 Credit transfer payment format for Sweden</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Radio frequency identifier**

Radio Frequency Identification (RFID) is a data-collection technology that uses electronic tags to store identification data and a no-line-of-sight requirement reader to capture the identification data.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Low customer usage and a limited feature set.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Inventory management</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics 365 for Operations 1611.</td>
</tr>
</tbody>
</table>

**Report about state invoices numbering for Latvia**

Latvian legislation provides specific rules about the numbering of sales invoices. The functionality lets you assign specific numbers to sales invoices, based on the user or user group. You can then generate a report or an XML file. You can also print a report about invoice numbers that are used.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The state invoice numbering no longer has to be maintained. The report about used invoice numbers is no longer required.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts receivable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Set up the names of the manager and general accountant of a company for Lithuania**
The names of the manager and the general accountant of a company can be specified in the company information and used in different local report printouts.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Replaced by another feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales and marketing, Inventory management</td>
<td>Yes, ISO20022 Credit transfer payment format and AvtaleGiro customer payment format for Norway, as well as pain.002 and camt.054 bank notification return files import.</td>
</tr>
<tr>
<td>Accounts payable, Accounts receivable</td>
<td>Yes, the setup of officials can be used for the same purpose.</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Shipping carrier interface**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Replaced by another feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duplicate functionality</td>
<td>Partially replaced by Transportation management</td>
</tr>
<tr>
<td>Sales and marketing, Inventory management</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics 365 for Operations version 1611.</td>
</tr>
</tbody>
</table>

**Telepay payment formats for Norway**

Telepay payment formats include vendor payment export (credit transfer) and customer payment collection (direct debit).

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Replaced by another feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>The payment formats are no longer used.</td>
<td>Yes, ISO20022 Credit transfer payment format and AvtaleGiro customer payment format for Norway, as well as pain.002 and camt.054 bank notification return files import.</td>
</tr>
<tr>
<td>Accounts payable, Accounts receivable</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Vendor payment export formats for Finland**

Two formats for exporting payments are available for Finland. LM02 (FI) is used for domestic payments, and LUM2 (FI) is used for foreign payments.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Replaced by another feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>The payment formats are no longer used.</td>
<td>Yes, ISO20022 Credit transfer payment format for Finland</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Warehouse management II**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The Warehouse management II solution (WMS II) that was available in the <strong>Inventory management</strong> module duplicates functionality that is in the <strong>Warehouse management</strong> module that was released in Dynamics AX 2012 R3.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>The <strong>Warehouse management</strong> module that was released in AX 2012 R3, Dynamics AX 2012 R3 CU8, and Dynamics AX 2012 R3 CU9 replaces the Warehouse management II features. The new module has more advanced features and more flexible warehouse management processes than Warehouse management II.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Inventory management, Sales and marketing, Procurement and sourcing</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics 365 for Operations version 1611</td>
</tr>
</tbody>
</table>

**Worker reminders in Human Resources**

**Human Resources Payroll information**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Low usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Human resources</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics 365 for Operations version 1611</td>
</tr>
</tbody>
</table>

**Workflow for creating goals**

A workflow for managing the creation of employee goals is one of several workflows that were available to help coordinate the performance management process.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Performance management has been completely redesigned in Finance and Operations.</th>
</tr>
</thead>
</table>
The redesigned Performance management feature gives more control over the content of the goals, the measurements that are used to track progress, and the attachment of supporting documentation. Goals can be stored as templates and then reused. This feature can help you set up additional goals for your employees more quickly.

**Product areas affected**
Human capital management

**Status**
Removed as of Dynamics 365 for Operations version 1611.

### Dynamics AX 7.0

#### Ability to cancel changes to a vendor invoice

**Reason for deprecation/removal**
Performance enhancement

**Replaced by another feature?**
No

**Product areas affected**
Accounts payable

**Status**
Removed as of Dynamics AX 7.0.

### AIF, AxD, and AxBC integrations

In Application Integration Framework (AIF), data can be exchanged with external systems through business logic that is exposed as services. Dynamics AX includes services that are based on documents and .NET Business Connector (AxBC). A document is created by using XML. The XML includes header information that is added to create a message that can be transferred into or out of Dynamics AX. Examples of documents include sales orders and purchase orders. However, almost any entity, such as a customer, can be represented by a document. Services that are based on documents use the **Axd <Document>** classes.

**Reason for deprecation/removal**
The architecture of AIF and AxDs could not be scaled to a cloud service. There were performance issues around bulk import.

**Replaced by another feature?**
This feature is replaced by the Data Import/Export framework, which supports recurring bulk import/export. For AxBC, we recommend that you use the actual tables.

**Product areas affected**
AxDs, AxBCs, and AIF

**Status**
Removed as of Dynamics AX 7.0.

### Billing code rate scripts

Billing scripts were used to calculate billing rates for billing codes. This scripts required custom development in the C Sharp or Visual Basic programming language. In the current version of Dynamics AX, the **billing code rate scripts** are not supported.
<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The support for the custom C Sharp or Visual Basic scripts was not added in Dynamics AX 7.0.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Public sector, Accounts receivable</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**BOMs without BOM versions**

When the **BOM versions** configuration key was disabled, bill of materials (BOM) versions were hidden in all forms, and the system forced a 1:1 relationship between released products and BOMs. In the current version of Dynamics AX, the **BOM versions** configuration key can't be disabled.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Using a configuration key to control BOM versions doesn't scale in a cloud environment.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Product information management, Inventory management</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Brazilian Bordero**

Specific method of payment for Brazilian companies

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Support for the Brazilian Bordero method of payment has been discontinued from Brazilian localization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Brazilian Sintegra statement**

Federal tax statement for ICMS tax

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>This statement is no longer applicable in some Brazilian states.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No. Users can use Generic Electronic reporting tool to configure the statement if required under specific situations.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Fiscal books</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Brazilian SCAN contingency mode for NF-e**

(SCAN) contingency environment is used to generate, export, and import the status of a Nota Fiscal eletrônica (NF-e) when the environment of Secretaria da Fazenda (SEFAZ) is not available.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>This method of contingency is no longer applicable in all Brazilian states</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts receivable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Business Analyzer**

This mobile application lets users review key business metrics.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>This functionality has been replaced by another feature.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>The Monitor financial performance content pack for Microsoft Power BI will include key financial metrics that were previously available in Business Analyzer.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>General ledger</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: The use of Business Analyzer has been deprecated.</td>
</tr>
</tbody>
</table>

**Business statistics**

The setup of business statistics inquiries that can help you analyze the performance of the organization.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Legacy approach to business intelligence (BI), low customer usage, and a limited feature set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>New BI solutions for the current version of Dynamics AX</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Procurement and sourcing, Accounts payable, Sales and marketing, Accounts receivable</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>
### Change document date function in Invoice approval journal

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Low usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes. The document date on the posted vendor transaction can be changed.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

### ClieOp03 payment format for the Netherlands

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The format is no longer applicable in the Netherlands, because it has been replaced by SEPA functionality.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>SEPA payments export</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

### Compliance Center

The Compliance Center was an Enterprise Portal site for managing the documentation requirements for compliance initiatives that are related to the Sarbanes-Oxley law.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Lack of customer usage. Microsoft SharePoint includes the same capability that was available in the Compliance Center.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Compliance and internal controls</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

### Connector for Microsoft Dynamics

This tool was used to integrate key data from Microsoft Dynamics CRM to Dynamics ERP applications.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>This functionality has been replaced by another feature.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Dataverse</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Connector for Dynamics</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td><strong>Container unit and multi dimension on-hand</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Reason for deprecation/removal</strong></td>
<td>Duplicate functionality</td>
</tr>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>Yes. Since AX 2012, this functionality has been replaced by the consolidated batch orders feature set. This feature set includes the consolidated on-hand view.</td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
<td>Product information management, Production control, Inventory management, Sales and marketing</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
<tr>
<td><strong>Cue group metadata</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Reason for deprecation/removal</strong></td>
<td>Cue groups were used to display one or more Cues in the FactBox area. There was limited uptake, and there were also performance concerns, because a record change in a parent form caused one query per Cue in the Cue group.</td>
</tr>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>No</td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
<td>All modules</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
<tr>
<td><strong>Cue metadata</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Reason for deprecation/removal</strong></td>
<td>Cue metadata was limited to count or sum information.</td>
</tr>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>Tile metadata was introduced to provide more flexibility for modeling. For example, you can model current counts, navigation, and key performance indicators (KPIs). Count tile metadata is the direct replacement of the Cue metadata.</td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
<td>All modules</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Removed as of Dynamics AX 7.0</td>
</tr>
<tr>
<td><strong>Danish check format</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Reason for deprecation/removal</strong></td>
<td>Support for the Danish check format layout has been discontinued, and the report has been removed from DK localization.</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>No</td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
<td>All modules</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Data partitions**

Data partitions provide a logical separation of data in the Dynamics AX database.

<table>
<thead>
<tr>
<th><strong>Reason for deprecation/removal</strong></th>
<th>Data partitions were introduced in Dynamics AX 2012 R2 to enable data isolation. In a common scenario, a company has subsidiaries, and the data from one subsidiary should not be visible to another subsidiary, even though both subsidiaries are managed by the same IT department. However, extra scripts and management overhead throughout the program were required in order to create new partitions and populate them with data, and to back up partition data. In the cloud, where we have access to platform as a service (PaaS) database services (Microsoft Azure SQL Database), it’s much more efficient to use a database as the isolation container than to do isolation in the program. Regardless of whether data partitioning is required for subsidiaries, for multiple tenants, or just for scale, we believe that the scenarios can be handled better through multiple instances of Finance and Operations.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>Customers using data partitions must use multiple instances of Finance and Operations if database level separation is a critical issue.</td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
<td>All modules</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Database and file share storage for attachments**

Dynamics AX 2012 allowed storage of attachments in the database and in file shares. Both of those options are no longer supported.
<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Files share storage is no longer supported because cloud-hosted environments cannot communicate with local file shares. Database storage has been deprecated in favor of Azure Blob storage. Azure Blob storage is equivalent to storage in the database, as documents can only be accessed through Finance and Operations client forms. This provides the added benefit of providing storage that doesn't negatively affect the performance of the database. Blob storage is the default storage mechanism for Document Management and works immediately.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Database storage has been deprecated in favor of Azure Blob storage.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Delimitation**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>No use of the functionality was found.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Time and attendance</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Desktop client**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The Dynamics AX client experience has been redesigned to improve usability across multiple platforms and devices.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>The new web client is based on the desktop Form metadata and programming model that have been modified to provide a rich web platform.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Direct database connection**

In Dynamics AX 2012 R3, Retail Modern POS could connect directly to the Channel DB in similar fashion to Enterprise POS. This was in addition to the standard communication method of Retail Modern POS communicating through Retail Server.
| Reason for deprecation/removal | Direct database connectivity required lower security protocols and was primarily used to achieve the highest levels of performance. Due to the performance and security enhancements that have occurred in Finance and Operations, this functionality now causes more issues than it solves. |
| Replaced by another feature? | No. Only standard Retail Server communication is now supported. |
| Product areas affected | Channel DB/Retail Modern POS |
| Status | Removed as of Dynamics AX 7.0. |

**Dutch SWIFT MT940**

| Reason for deprecation/removal | Generic functionality is now used instead of localized functionality. |
| Replaced by another feature? | Yes, this functionality has been replaced by Advanced bank reconciliation functionality. |
| Product areas affected | All modules |
| Status | Deprecated: A removal date has not been set for this feature. |

**eBilanz (XBRL for Germany)**

This functionality provided eXtensible Business Reporting Language (XBRL) output that is intended specifically for the German eBilanz taxonomy.

| Reason for deprecation/removal | Lack of customer usage |
| Replaced by another feature? | This feature hasn't been replaced by another feature, but multiple specialized XBRL packages that provide rich XBRL functionality are available for the German market. |
| Product areas affected | Management Reporter |
| Status | Deprecated: A removal date has not been set for this feature. |

**Enterprise Portal client**

| Reason for deprecation/removal | A single client platform has been provided. |
### Replaced by another feature?

**The new web client is based on the desktop form metadata and programming model that have been modified to provide a rich web platform.**

<table>
<thead>
<tr>
<th>Product areas affected</th>
<th>All modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

### Environmental sustainability

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Low customer usage and a limited feature set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Compliance and internal controls, Accounts payable</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

### Form ActiveX and Managed Host controls

| Reason for deprecation/removal | The ActiveX and Managed Host controls are based on the deprecated desktop client. |
| Replaced by another feature?   | The extensible control framework supports building new controls that are based on HTML, CSS, and JavaScript, and is a first-class control in the Microsoft Visual Studio Tooling environment. |
| Product areas affected         | All modules |
| Status                         | Removed as of Dynamics AX 7.0. |

### Generate prenotes by using a batch

Prenote generation can't be done by using a batch, but it can still be done by a user.

| Reason for deprecation/removal | No form exists to persist and display the resulting prenote file when it's generated by using a batch. |
| Replaced by another feature?   | Prenotes can still be generated, and the user has control over the location where the file is saved. |
| Product areas affected         | Accounts payable, Accounts receivable, Cash and bank management |
| Status                         | Removed as of AX 7.0. |

### German DTAUS payment export and account statement import (totals and transactions)

Prenote generation can't be done by using a batch, but it can still be done by a user.
<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The format is no longer applicable in Germany, because it has been replaced by Single Euro Payments Area (SEPA) functionality.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, this functionality has been replaced by SEPA payment export and advanced bank reconciliation functionality for importing account statements.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**German DTAZV payment format in domestic Currency**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The format is no longer applicable in Germany, because it has been replaced by SEPA functionality.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>SEPA payments export</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**German MT940 import**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Generic functionality is now used instead of localized functionality.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, this functionality has been replaced by Advanced bank reconciliation functionality.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**German XML EU Sales list**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The XML format for German EU Sales List reporting is no longer supported. Only the ELMA5 text file format can be used to submit the EU Sales List report to the German Tax Office.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Tax</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**GL SSRS reports**

Reports that include the following menu items have been removed: **Summary trial balance, Detailed trial balance, Chart of accounts, Audit trail, Balances,** and **Balance list.**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Financial Microsoft SQL Server Reporting Services (SSRS) reports have been replaced by Management Reporter capabilities and default reports.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Management Reporter (labeled Financial reporting in the current version of Dynamics AX)</td>
</tr>
</tbody>
</table>

**InfoPart and FormPart metadata**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>InfoPart and FormPart metadata enabled the creation of FactBoxes for two different clients.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>InfoPart metadata, which was a simplified form definition, is converted into a Form by upgrade tooling. FormPart metadata, which referenced a Form, is replaced by a more direct reference that is created by upgrade tooling.</td>
</tr>
</tbody>
</table>

**Main account list page**

A list of accounts for the legal entity and related balance information

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Balance information is available on the Trial balance list page by account and dimension.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td><strong>Main accounts</strong> contains the same list of accounts that the <strong>Main account list page</strong> contained. The grid view in <strong>Main accounts</strong> also shows an even smaller, grid-like view.</td>
</tr>
</tbody>
</table>

**Product areas affected**

**Product areas affected**

**Status**

**Status**

**Product areas affected**

**Status**
<table>
<thead>
<tr>
<th><strong>Malaysia and Singapore bank cash flow report</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>This feature let the user print a cash flow report that shows transactions and details of the cash inflows and outflows for a specific date range for selected bank accounts.</td>
</tr>
<tr>
<td><strong>Reason for deprecation/removal</strong></td>
</tr>
<tr>
<td><strong>Replaced by another feature?</strong></td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
</tr>
<tr>
<td><strong>Status</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Mexican CFD electronic invoice</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>This feature enabled the generation of Mexican electronic invoices by using the Comprobante Fiscal Digital (CFD) method, where the company signs the invoice by requesting the related authorization from the government. This feature also provides a monthly report that includes all electronic invoices that were issued in the period.</td>
</tr>
<tr>
<td><strong>Reason for deprecation/removal</strong></td>
</tr>
<tr>
<td><strong>Replaced by another feature?</strong></td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
</tr>
<tr>
<td><strong>Status</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Mexico realized and unrealized VAT</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamics AX 2012 managed unrealized value-added tax (VAT) by using Mexico-specific functionality for unrealized tax.</td>
</tr>
<tr>
<td><strong>Reason for deprecation/removal</strong></td>
</tr>
<tr>
<td><strong>Replaced by another feature?</strong></td>
</tr>
<tr>
<td>Product areas affected</td>
</tr>
<tr>
<td>----------------------------------------</td>
</tr>
<tr>
<td>Status</td>
</tr>
</tbody>
</table>

**Microsoft Outlook integration**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>This functionality has been replaced by Microsoft Exchange Server integration.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Sales and marketing</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Private blocking of inventory and warehouse management journals**

The inventory and warehouse journals no longer support the ability to mark a journal as private for a selected user. Only the process of blocking journals as private for user groups and blocking during editing is supported.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>No use of the functionality was found.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Inventory management</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Product builder**

Product builder was used to dynamically configure items from a sales order, purchase order, production order, sales quotation, project quotation, or item requirement. Based on a product model that had modeling variables, the user could select values to meet the customer requirements and get a unique product variant that had a BOM and route.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Product builder exposed X++ code to end users and isn't supported in the current version of Dynamics AX. It has been removed to avoid duplicate maintenance efforts on overlapping, sizeable codebases.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes. The constraint-based configuration was introduced in Dynamics AX 2012 where the depreciation of Product builder in future versions was already announced. The constraint-based configuration technology is selected on the product masters to enable the configuration. To learn more, see Product configuration overview.</td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
<td>Product information management, Sales and marketing</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Production Floor app**

This is the app for tablet devices running Windows 8.1 RT and Windows 8.1 Pro.

<table>
<thead>
<tr>
<th><strong>Reason for deprecation/removal</strong></th>
<th>With the change to a web-based client, it is possible to deliver similar functionality through the native Dynamics AX 7.0 client. The Job Card Device provides a production floor user interface that is optimized for touch and tablet form factors.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>Yes. The Job Card Device, which is a native part of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Product areas affected</strong></th>
<th>Production control</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status</strong></td>
<td>Deprecated: A removal date from the Microsoft store has not yet been set for this feature.</td>
</tr>
</tbody>
</table>

**Rename product dimension**

This feature let you change the name of one of the three standard product dimensions (size, color, or style) to a name that better suited your business requirements. Renaming included all the labels where the product dimension name was used.

<table>
<thead>
<tr>
<th><strong>Reason for deprecation/removal</strong></th>
<th>The current version of Dynamics AX doesn't support label changes at run time.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Product areas affected</strong></th>
<th>Product information management</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status</strong></td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Retail Server connectivity using HTTP**

In Dynamics AX 2012 R3, the Retail Server could function using HTTP communication (non-secured). This was in addition to the standard communication using HTTPS.

<table>
<thead>
<tr>
<th><strong>Reason for deprecation/removal</strong></th>
<th>Due to new security requirements, only secured communication using TLS 1.2 (or above, as available) is now supported. The self-service installer will automatically configure the computer for this communication.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>No. Only standard HTTPS communication is now supported.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Retail Server</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Role Center pages**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Role Center pages were built on the deprecated Enterprise Portal platform, which has been replaced by the new web client platform in the current version of Dynamics AX.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>The new Workspace form pattern provides users with a process-centered design that provides easy access to commonly used tasks within that process.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0</td>
</tr>
</tbody>
</table>

**Sales tax jurisdictions**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Low customer usage and a limited feature set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>US sales tax</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Sites Services**

Sites Services let you build websites that extend your business processes to the Internet without IT support.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The Microsoft Azure infrastructure that is used by Dynamics AX has new capabilities that can be used instead (for example, Azure sites).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>HR recruiting, Case management, Request for quotes, Vendor registration, Collaborative workspaces for opportunities and campaigns</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**SSAS demand forecasting strategy**
<table>
<thead>
<tr>
<th><strong>Reason for deprecation/removal</strong></th>
<th>The design of the feature cannot be supported in the new cloud architecture.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>Azure Machine Learning demand forecasting strategy</td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
<td>Master planning</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Vendor invoice pool excluding posting details**

<table>
<thead>
<tr>
<th><strong>Reason for deprecation/removal</strong></th>
<th>Low usage. This functionality has been replaced by the Invoice journal that has workflow functionality.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>Workflow capabilities of the Invoice journal.</td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
<td>Accounts payable</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Virtual company accounts**

The virtual companies feature is no longer supported in Dynamics AX. The virtual companies feature let users set up tables that could be shared by a set of companies. For a description of the feature, see Company accounts and Virtual company accounts. The feature works by grouping tables into collections that are assigned to virtual companies, which are groups of existing “real” companies. Queries are created so that all the companies in the virtual company can access the data in the tables of the associated table collections.

<table>
<thead>
<tr>
<th><strong>Reason for deprecation/removal</strong></th>
<th>- Virtual companies must be set up before data is stored in the tables. Retrofitting virtual companies onto an existing implementation is very difficult.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Because there has been so much data normalization in the current version of Dynamics AX, it has become difficult to know what to add to the table collections. For example, it's difficult to know which tables to share. All the tables referenced from tables that are in a virtual company must also added. Because of table normalization, even simple master data that is spread across multiple tables must be part of the virtual company. Any mistake that is made here will cause functional issues.</td>
</tr>
<tr>
<td></td>
<td>- When a table is part of a virtual company, it loses information about the origin of the data, and only the virtual company is recorded.</td>
</tr>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>Global tables can be used to make tables accessible from all companies. Currently, there is no replacement.</td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

**Windows 8 tablet app**

The Windows 8 tablet app provided functionality for expense entry and approval.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Finance and Operations is compatible with tablets. The tablet app is no longer required.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Expense management</td>
</tr>
<tr>
<td>Status</td>
<td>Removed: This functionality is only available for Dynamics AX 2012 R3.</td>
</tr>
</tbody>
</table>

**Workplanner**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Low usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No, but the Profile relation page, which is opened from the Profile groups page, supports the same business scenario as the deprecated Workplanner page.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Time and attendance</td>
</tr>
<tr>
<td>Status</td>
<td>The code has not been removed. However, the form, JmgWorkPlanner, was not migrated.</td>
</tr>
</tbody>
</table>

**X++ financial statements**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>This functionality has been replaced by another feature.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Management Reporter (labeled Financial reporting in the current version of Dynamics AX)</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>General ledger</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 2012</td>
</tr>
</tbody>
</table>
This topic lists features of Microsoft Dynamics AX 2012 that were postponed. These features weren't implemented in Microsoft Dynamics AX 7.0. In the following table, the **Current status** column indicates whether the feature has been implemented since the AX 7.0 release.

For a detailed list of the release date for each version, see [Software lifecycle policy and cloud releases](#).

<table>
<thead>
<tr>
<th>AX 2012 feature that was postponed</th>
<th>Description</th>
<th>Current status (as of February 2019)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absence management in Human resources</td>
<td>Functionality for entering absence transactions isn't included. Additionally, functionality for approving absence transactions as a manager isn't included. Setup capabilities that are required for integration with other modules are available through the Human Resources 2 configuration key.</td>
<td>Implemented in Dynamics 365 Human Resources</td>
</tr>
<tr>
<td>Alerts</td>
<td>Alerts help users keep track of data changes in the system.</td>
<td>Implemented in Platform update 15</td>
</tr>
<tr>
<td>Bank payment order for Latvia and Lithuania</td>
<td>You can print a payment order for Latvia and Lithuania. This feature will be available in a future update.</td>
<td>Not implemented</td>
</tr>
<tr>
<td>Bankgirot AP return format for Sweden</td>
<td>The Bankgirot return format is used to import bank return messages. This feature will be available in a future update.</td>
<td>Not implemented</td>
</tr>
<tr>
<td>Client drag-and-drop</td>
<td>The web client controls have application programming interfaces (APIs) for drag-and-drop operations, but these APIs are based on the deprecated desktop client technology and they require a redesign so that they work on the new web client platform. APIs that support drag-and-drop operations will be reviewed for inclusion in a future update.</td>
<td>Not implemented</td>
</tr>
<tr>
<td>Client right-to-left (RTL) layout</td>
<td>RTL layout is now supported.</td>
<td>Implemented in Platform update 2</td>
</tr>
<tr>
<td>AX 2012 Feature That Was Postponed</td>
<td>Description</td>
<td>Current Status (as of February 2019)</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>Cost accounting</td>
<td>The <strong>Cost accounting</strong> module is designed to meet the requirements of internal costs and profitability reports at multiple organizational levels. To define the cost object level, the module depends on a correct mapping of financial dimensions. The module lets you perform advanced allocations of cost origin from expenditures that are registered in the general ledger or budget. It also lets you compare realized costs and budgeted costs.</td>
<td>Implemented in version 1611</td>
</tr>
<tr>
<td>Customer self-service (CSS)</td>
<td>CSS lets you create approved customer records. It also allows users to view selected product catalogs, order items, and view the status of invoices. Additionally, CSS lets you create and follow return orders.</td>
<td>Not implemented</td>
</tr>
<tr>
<td>Customizable help topics</td>
<td>The ability to create customized help topics has not yet been implemented. Custom task guides and custom field help are available. This feature will be available in a future update.</td>
<td>Not implemented</td>
</tr>
<tr>
<td>Employee self-service (ESS)</td>
<td>ESS shows employees several tiles that have task-related and career-related information on a single page. Employees can view pending work items and click links that open pages where they can take action on their tasks. ESS pages also show employees the status of their certifications, when their next performance reviews are scheduled, skills, goals, and compensation information, and other information, such as balances for vacation and sick time. Employees can also access a company directory from their ESS page.</td>
<td>Implemented in version 1611</td>
</tr>
<tr>
<td>External questionnaire and recruiting functionality</td>
<td>Functionality for externally posting questionnaires and open jobs will be added to Human Resources in a future update.</td>
<td>External questionnaire functionality hasn't been implemented. Recruiting functionality is available in Microsoft Dynamics 365 Talent: Attract.</td>
</tr>
<tr>
<td>Fiscal printers for Poland</td>
<td>Integration with Polish fiscal printers enables the required information to be sent to the fiscal printer in the correct format during invoice posting. Examples of Polish fiscal printers include the Posnet Thermal and Elzab Omega printer types. This feature will be available in a future update.</td>
<td>Not implemented</td>
</tr>
<tr>
<td>AX 2012 FEATURE THAT WAS POSTPONED</td>
<td>DESCRIPTION</td>
<td>CURRENT STATUS (AS OF FEBRUARY 2019)</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>General budget reservations</td>
<td>The General budget reservations document is sometimes referred to as a commitment. Public sector entities often use this document to set aside or earmark budgeted funds so that they aren't available for other purposes.</td>
<td>Implemented in version 8.1</td>
</tr>
<tr>
<td>Graphics tab on the Fixed asset value model and Depreciation book profile pages</td>
<td>The chart shows the depreciation, accumulated depreciation, and net book value over time. Users can click the Data tab to view more detailed information than the chart shows. This chart will be redesigned in a future update.</td>
<td>Not implemented</td>
</tr>
<tr>
<td>Intelligent Data Management Framework (IDMF)</td>
<td>IDMF is an add-on tool that lets system administrators optimize performance. IDMF assesses the health of the application, analyzes current usage patterns, and helps reduce database size.</td>
<td>Not implemented</td>
</tr>
<tr>
<td>Microsoft Project client integration</td>
<td>The Microsoft Project client is integrated with projects.</td>
<td>Implemented in version 7.2 (July 2017 update)</td>
</tr>
<tr>
<td>Procurement site</td>
<td>In previous versions, the Employee self-service procurement site lets you enter requisitions for employees, view the status of an order (created, received, or receipt confirmed), and request onboarding of a new vendor. You could configure different procurement catalogs to show on the site depending on policy. You could also design procurement catalogs by adding new nodes. In the current version, procurement catalog capabilities are reduced and are used only to limit the products that can be ordered for an organization. The structure is always based on the Procurement categories hierarchy. Additionally, on the procurement site the employee could approve a vendor invoice and confirm receipts in relation to the requisitions and derived purchase orders.</td>
<td>Not implemented</td>
</tr>
<tr>
<td>Secure global address book</td>
<td>The ability to help secure the global address book by legal entity and address book is not available. This feature will be available in a future update.</td>
<td>Not implemented</td>
</tr>
<tr>
<td>Specifications for Electronic reporting (ER) payment formats</td>
<td>Currently, you must enter the payment format specifications manually. In a future update, you will be able to select payment format specifications in</td>
<td>Not implemented</td>
</tr>
</tbody>
</table>
The following payment specifications are currently supported per payment format. [NOTE] Values for these supported payment specifications are used as payment specification parameters on the Payment specification page for a selected method of payment.

### BTL91 for the Netherlands

<table>
<thead>
<tr>
<th>PAYMENT SPECIFICATION (USED IN ER)</th>
<th>EXPORT FORMAT DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChqBen</td>
<td>Cheque, Begunstigde</td>
</tr>
<tr>
<td>ChqOff</td>
<td>Cheque, Kantoor opdrachtgever</td>
</tr>
<tr>
<td>ChqPri</td>
<td>Cheque, Opdrachtgever</td>
</tr>
<tr>
<td>TrfBenBen</td>
<td>Overboeking Begunstigde/Be gunstigde</td>
</tr>
<tr>
<td>TrfBenBenUrg</td>
<td>Overboeking Begunstigde/Be gunstigde Spoed</td>
</tr>
<tr>
<td>TrfEurBen</td>
<td>Overboeking Euro/Begunstigde</td>
</tr>
<tr>
<td>TrfEurBenUrg</td>
<td>Overboeking Euro/Begunstigde Spoed</td>
</tr>
<tr>
<td>TrfEurEur</td>
<td>Overboeking Euro/Euro</td>
</tr>
<tr>
<td>TrfEurEurUrg</td>
<td>Overboeking Euro/Euro Spoed</td>
</tr>
<tr>
<td>TrfforBen</td>
<td>Overboeking VV-rekening/Begunstigde</td>
</tr>
<tr>
<td>TrfForBenUrg</td>
<td>Overboeking VV-rekening/Begunstigde Spoed</td>
</tr>
<tr>
<td>TrfforFor</td>
<td>Overboeking VV-rekening/VV-rekening</td>
</tr>
<tr>
<td>AX 2012 FEATURE THAT WAS POSTPONED</td>
<td>PAYMENT SPECIFICATION (USED IN ER)</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>TrfForForUrg</td>
<td>Overboeking VV-rekening/VV-rekening Spoed</td>
</tr>
<tr>
<td>Betalingsservice for Denmark</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B0112</td>
</tr>
<tr>
<td></td>
<td>B0113</td>
</tr>
<tr>
<td></td>
<td>T0112</td>
</tr>
<tr>
<td></td>
<td>T0117</td>
</tr>
<tr>
<td>Nordea vendor for Denmark</td>
<td></td>
</tr>
<tr>
<td></td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>52</td>
</tr>
</tbody>
</table>
### AX 2012 Feature That Was Postponed

<table>
<thead>
<tr>
<th>AX 2012 Feature That Was Postponed</th>
<th>Payment Specification (Used in ER)</th>
<th>Export Format Description</th>
<th>Current Status (As of February 2019)</th>
</tr>
</thead>
<tbody>
<tr>
<td>43 Request for transfer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46 Transfer form/giro payment</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### ISO20022 Credit Transfer (CH)

<table>
<thead>
<tr>
<th>Payment Specification (Used in ER)</th>
<th>Export Format Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tp1.ESROPS</td>
<td>Type 1 - ESR orange payment slip</td>
<td></td>
</tr>
<tr>
<td>Tp21.ISR1SPS</td>
<td>Type 2.1 - IS red 1 stage payment slip</td>
<td></td>
</tr>
<tr>
<td>Tp22.ISR2SPS</td>
<td>Type 2.2 - IS red 2 stage payment slip</td>
<td></td>
</tr>
<tr>
<td>Tp7.Dmstc</td>
<td>Type 7 - Domestic postal order</td>
<td></td>
</tr>
<tr>
<td>TpE1.PSWR</td>
<td>Type E1 - Payment slip with reference</td>
<td></td>
</tr>
<tr>
<td>TpE2.PSWN</td>
<td>Type E2 - Payment slip with notifications</td>
<td></td>
</tr>
</tbody>
</table>

### AvtaleGiro (NO)

<table>
<thead>
<tr>
<th>Payment Specification (Used in ER)</th>
<th>Export Format Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Varsling</td>
<td>AvtaleGiro-trans with notification</td>
<td></td>
</tr>
</tbody>
</table>

### AutoGiro (NO)

<table>
<thead>
<tr>
<th>Payment Specification (Used in ER)</th>
<th>Export Format Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Melding</td>
<td>Autogiro-trans with notification</td>
<td></td>
</tr>
</tbody>
</table>

### eFaktura (NO)
<table>
<thead>
<tr>
<th>PAYMENT SPECIFICATION (USED IN ERP)</th>
<th>EXPORT FORMAT DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reklame</td>
<td>Include advertising flag</td>
</tr>
</tbody>
</table>

**ISO20022 Credit transfer (DK)**

<table>
<thead>
<tr>
<th>PAYMENT SPECIFICATION (USED IN ERP)</th>
<th>EXPORT FORMAT DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>EasyAccountTransfer</td>
<td>Easy-account with CVR (NKV)</td>
</tr>
<tr>
<td>Paym_slip</td>
<td>Transfer forms (OCR)</td>
</tr>
</tbody>
</table>

**ISPAG-CNAB240 format (BR)**

<table>
<thead>
<tr>
<th>PAYMENT SPECIFICATION (USED IN ERP)</th>
<th>EXPORT FORMAT DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>OP (payment order), DOC (wire transfer), TED (other type of wire transfer), and direct credit in the account</td>
</tr>
<tr>
<td>J</td>
<td>Bar code payments (invoice with bar code or other type of documents with bar code)</td>
</tr>
<tr>
<td>O</td>
<td>Tax payments or other public services payments</td>
</tr>
</tbody>
</table>

**US Payroll**

US Payroll provides gross-to-net processing for employees in the United States. In Payroll, you can set up, enter, and maintain all payroll records and transactions.

Implemented in version 1611
<table>
<thead>
<tr>
<th>AX 2012 FEATURE THAT WAS POSTPONED</th>
<th>DESCRIPTION</th>
<th>CURRENT STATUS (AS OF FEBRUARY 2019)</th>
</tr>
</thead>
<tbody>
<tr>
<td>vendor collaboration (Vendor Portal)</td>
<td>Dynamics AX 2012 provided vendor portal capabilities via Enterprise Portal. Financial and Operations also provides these capabilities. In version 7.1 (also known as Dynamics 365 for Operations 1611), a vendor could view and respond to purchase orders. In version 7.3, the vendor can view and respond to RFQs. Vendors can also view and edit selected information from the vendor record such as addresses, contact information, and contact persons, and they can upload documents in relation to their certifications.</td>
<td>Implemented in version 7.3</td>
</tr>
<tr>
<td>vendor requests - external request to become a new vendor</td>
<td>Dynamics AX 2012 provided the ability for an anonymous user to sign up to be a vendor in the system, which could lead to a vendor request for adding a new vendor to the vendor master. In version 7.3, the anonymous request from a prospective vendor can be imported via an entity (Data Management/OData), which can lead to inviting the vendor - or the vendor's contact person - to register more details about the prospective vendor. The information provided is included in a new vendor request that can be reviewed and approved via a workflow process. An approval of the vendor request leads to creation of a new vendor account.</td>
<td>Implemented in version 7.3</td>
</tr>
<tr>
<td>vendor requests in general</td>
<td>Dynamics AX 2012 had a concept of vendor requests that served various purposes related to updating vendor-related information, such as requesting new procurement categories for the vendor, internal employees requesting new vendors, or requesting to add a vendor to another company. Only the vendor's request of being added as a vendor has been implemented in version 7.3.</td>
<td>Not implemented</td>
</tr>
<tr>
<td>AX 2012 Feature that was postponed</td>
<td>Description</td>
<td>Current status (as of February 2019)</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-------------</td>
<td>--------------------------------------</td>
</tr>
</tbody>
</table>
| **[Russia] Tax registers**        | Legal entities can use registers to disclose their revenues and expenses. The registers are used to track revenue and expense data from the time that primary documents, such as sales invoices and delivery notes, are first entered by using the calculation of cost prices for production. The data from the registers is used to confirm the declared profit of the legal entity. This functionality includes the following features:  
  - Current period incomes  
  - Tax expenses  
  - Other expenses of current period  
  - Unrealized expenses of current period  
  - Other unrealized expenses  
  - Accounts receivable debt – inventory  
  - Bad debts reserve calculation  
  - Bad debts reserve movement  
  - Accounts receivable movement  
  - Procedure for writing-off AR bad debts  
  - Accounts payable debt - inventory  
  - Accounts payable debt movement  
  - Procedure for writing-off AP bad debts  
  - Goods cost calculation  
  - FA object information  
  - IA object information  
  - FA depreciation  
  - IA depreciation  
  - FA/IA sale  
  - Depreciation bonus recovery | Implemented in version 8.1.3 |
<p>| <strong>[Russia] Electronic export/import format for Client-Bank interface and reconciliation procedure</strong> | Electronic formats for export of outgoing payments, and import of incoming payments. | Implemented in version 8.1.3 |
| <strong>[Russia] VAT declaration</strong>      | Electronic format of VAT declaration. | Implemented in version 10.0.1 |
| <strong>[Russia] Cash Flow Management</strong> | The functionality which obtains a cash flow forecast and performs an analysis, manages payments on a daily basis using payment schedule journals, controls the company's cash position, and maintains the company's cash flows with centralized control, | Implemented in version 10.0.1 |</p>
<table>
<thead>
<tr>
<th>AX 2012 FEATURE THAT WAS POSTPONED</th>
<th>DESCRIPTION</th>
<th>CURRENT STATUS (AS OF FEBRUARY 2019)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Russia] Accounting reporting format</td>
<td>Electronic format of the following accounting reports: BalanceSheet, IncomeStatement, CashFlow, EquityStatement, TargetUsageMoney</td>
<td>Implemented in version 10.0.1</td>
</tr>
<tr>
<td>[Russia] Assessed tax reporting</td>
<td>Assessed tax declaration.</td>
<td>Implemented in version 10.0.1</td>
</tr>
<tr>
<td>[Russia] Land tax reporting</td>
<td>Land tax declaration. Creation of Land tax declaration by separate divisions.</td>
<td>Implemented in version 10.0.1</td>
</tr>
<tr>
<td>[Russia] Transport tax reporting</td>
<td>Transport tax declaration.</td>
<td>Implemented in version 10.0.1</td>
</tr>
<tr>
<td>[Russia] Indirect tax return (VAT and Excise) on import of goods</td>
<td>Indirect (withholding) tax return (VAT and Excise) on import of goods from state members of Customs union.</td>
<td>Implemented in version 10.0.1</td>
</tr>
<tr>
<td>[Russia] Journal of Alcohol sales in Retail</td>
<td>Daily Alcohol journal.</td>
<td>Implemented in version 10.0.1</td>
</tr>
<tr>
<td>[Russia] Optional posting of transfer orders to General ledger</td>
<td>Option to post/not post transactions to General ledger when posting a transfer order.</td>
<td>Implemented in version 8.1.2</td>
</tr>
<tr>
<td>[Russia] Inventory owner</td>
<td>Inventory dimension used to track owner of inventory (consignment stock, bailment, tolling, etc.).</td>
<td>Implemented in version 10.0.1</td>
</tr>
<tr>
<td>[Russia] AP/AR - Third-party miscellaneous charges</td>
<td>Registration of third-party miscellaneous charges and allocation by the following regimes: Inclusion into cost of purchased goods (allocation to invoices lines from other vendors), and redrawing to other parties re-allocation to other expense accounts.</td>
<td>Implemented in version 8.1.1</td>
</tr>
<tr>
<td>[Russia] Goods in transit from vendor</td>
<td>Registering goods in transit from vendor by special posting profile with Item type &quot;purchased items en route&quot;. Creating Act of inventory holdings en route. (INV-6)</td>
<td>Implemented in version 8.1.2</td>
</tr>
<tr>
<td>[Russia] Goods in transit - sales to customer with postponed passing of property</td>
<td>Post sales invoice with postponed property transfer: no customer debts posted, all outgoing taxes are posted, items are transferred to transit warehouse. Register passing of property with posting debts and items sale from transit warehouse.</td>
<td>Implemented in version 8.1.2</td>
</tr>
<tr>
<td>[Russia] Bailment - accounting at bailee side</td>
<td>Accounting of inventory receipt for bailment as required by the Law and generation of primary form MX-1. Accounting of inventory return from bailment and generation of primary form MX-3. Bailment costs calculation from bailee side.</td>
<td>Implemented in version 8.1.2</td>
</tr>
<tr>
<td>AX 2012 FEATURE THAT WAS POSTPONED</td>
<td>DESCRIPTION</td>
<td>CURRENT STATUS (AS OF FEBRUARY 2019)</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>[Russia] Bailment - accounting at owner side</td>
<td>Accounting of inventory transfer to bailment and inventory return from bailment on goods owner side under bailment service contract.</td>
<td>Implemented in version 8.1.2</td>
</tr>
<tr>
<td>[Russia] Localization of Process Industries solution</td>
<td>Basic localization in two areas: correspondence of accounts for all new general ledger postings, and functional coexistence of Process Industries features and Russian country context (no issues when both Process Industries and Russian country context are enabled).</td>
<td>Implemented in version 10.0.1</td>
</tr>
<tr>
<td>[Russia] Alcohol sales declarations: Application 6, 7, 8 for wholesale. Applications 11, 12 for retail</td>
<td>Keeping track of alcoholic beverages types including producers, unit of measures, licenses for retail and wholesale trade. Preparing data for alcoholic beverages activities, including printing declarations and exporting them in XML format through e-reporting.</td>
<td>Implemented in version 10.0.1</td>
</tr>
</tbody>
</table>
Mainstream support for Dynamics AX 2009 Service Pack 1 (SP1), Dynamics AX 2012, and Dynamics AX 2012 R2 ended on October 9, 2018. After that date, only security hotfixes will continue to be provided for these three versions through the extended support period that continues until April 12, 2022. For more information, see support.microsoft.com.

Mainstream support for Dynamics AX 2012 R3 ended on October 12, 2021. After that date, only security hotfixes will continue to be provided through the extended support period that continues until January 10, 2023. For more information, see support.microsoft.com.

Customers are advised to upgrade to the latest version of Finance and Operations apps, such as Dynamics 365 Finance, Supply Chain Management, Commerce, and Project Operations:

- Dynamics AX 2009 Service Pack 1 customers should use the migration tool that is available.
- Dynamics AX 2012 and Dynamics AX 2012 R2 customers should upgrade to Finance and Operations apps through Dynamics AX 2012 R3 using the upgrade tool that is available. For additional upgrade information, see Upgrade from AX 2012 to Finance and Operations apps.

Frequently asked questions

When does the mainstream support for Dynamics AX 2009 Service Pack 1, Dynamics AX 2012, and Dynamics AX 2012 R2 end?
Mainstream support ended on October 9, 2018.

When does the mainstream support for Dynamics AX 2012 R3 end?
Mainstream support ended on October 12, 2021.

Was the information of the end date of the mainstream support for Dynamics AX 2009 Service Pack 1, Dynamics AX 2012, Dynamics AX 2012 R2 and Dynamics AX 2012 R3 available before?
Yes, it was always publicly available on the Microsoft Support Lifecycle site at support.microsoft.com.

Can customers on Premier Extended Hotfix Support or on Unified Support Advanced and Performance Levels get a non-security hotfix or regulatory update?
No. Neither non-security hotfixes nor regulatory updates will be available for the Dynamics AX products during the Extended Support phase of the product lifecycle (Dynamics AX 2009 SP1, Dynamics AX 2012, Dynamics AX 2012 R2, or Dynamics AX 2012 R3).

While the ability to request a non-security hotfix for select products is included with Unified Support Advanced and Performance Levels, Microsoft has determined that non-security hotfixes cannot be provided with a commercially reasonable effort for these products. As a result, no requests for non-security hotfixes or regulatory updates will be accepted.

I knew about the regulatory change before October 9, 2018, but it has the law enforcement date after October 9, 2018. Will I still get a regulatory update for Dynamics AX 2009 Service Pack 1, Dynamics AX 2012, and Dynamics AX 2012 R2?
No, we will only provide regulatory updates for Dynamics AX 2009 Service Pack 1, Dynamics AX 2012, and
Dynamics AX 2012 R2 for regulatory changes with the law enforcement dates on or earlier than October 9, 2018.

A customer or partner can already download a fix through LCS and inspect the code by installing it into a test Dynamics AX 2012 R3 environment. Is there any difference with the approach that you have proposed?

No, there is no difference.

How are binary hotfixes handled for Dynamics AX 2009 Service Pack 1, Dynamics AX 2012, Dynamics AX 2012 R2, and Dynamics AX 2012 R3?

If a hotfix is needed for a part of the system where Microsoft does not provide the source code and it is not a security bug, the hotfix will not be provided.

Can I access information about upcoming legislation changes in supported countries?

Yes, all legislation changes regardless of source (such as vendors, Microsoft research, or the localization community) are stored in the Lifecycle Service (LCS) alerting project. To access the LCS alerting project, follow these steps:

1. **Sign up**: Send an email request to join the localization community (under NDA) at DynRegW@microsoft.com.
2. **Access**: Sign in to the LCS project Regulatory Alerts - Worldwide (available only for companies with NDA or individuals who signed up for the Insider Program).
3. **Alerting guide**: Inform Microsoft about country/region regulation alerts and track the status of regulatory features. For more information, see [Submit alerts about country/region-specific regulatory features](#).

Note that the LCS alerting project includes all identified/reported legislation changes (alerts) but not actual Microsoft plans.

Can I get visibility into Microsoft plans for releases of new regulatory features?

Yes, in LCS Issue Search, information is published regarding upcoming regulatory updates for Dynamics 365 Finance, Supply Chain Management, Commerce, and Project Operations.

Are details about released regulatory features available?

Yes, for larger features, we publish this information in [Globalization resources](#).

Can I view and try actual regulatory features in Dynamics 365 prior to a release?

Yes, you can do this if you have purchased cloud licenses for Dynamics 365 and you have access to the Preview Early Access Program (PEAP). For regulatory changes specified by authorities early, the features will be available in PEAP prior to the law-enforced date (or latest date for first filing).

Can I access the code and configurations for regulatory features prior to a release?

Yes, the code will be available in the LCS project PEAP Assets, in the Shared asset library under Software deployable package. To sign up for the PEAP program, complete the [Preview Early Access Program for Finance and Operations applications](#) survey.

Configurations are available in the Global repository. For more information, see [Download ER configurations from the Global repository of Configuration service](#).
These topics describe tools and guidelines related to continuous delivery of your solution.

FAQ and guidelines

- Development and continuous delivery FAQ

Build and test automation

- Deploy and use an environment that supports continuous build and test automation
- Build automation that uses Microsoft-hosted agents and Azure Pipelines
- Testing and validations
- Integrate the POS with a new hardware device and generate the extension installer
- SysTest filtering using class and method attributes
- Acceptance test library resources

Advanced topics in build automation

- Exclude test packages from build output
- Manage third-party models and runtime packages by using source control
- Update model versions in the automated build

Blogs

- Insider tips on development and customization (blog)

Servicing

- Download updates from Lifecycle Services (LCS)
- Apply updates to cloud environments
- Install metadata hotfixes in development environments
- Patch SQL Server Reporting Services (SSRS) in one-box environments
- Update the Visual Studio development tools
This topic summarizes answers to questions that are frequently asked by ISVs and partners, especially regarding guidelines about development, testing, delivery, and lifecycle management.

Customization

Do I customize (overlayer) or use extensions?
Extensibility is the only customization framework in Finance, Supply Chain, and Commerce. Overlaying isn’t supported.

Dynamics 365 Finance, Supply Chain, and Commerce are extensively customized by partners, value added resellers (VARs), and even some customers. The ability to customize the product is a strength that has historically been supported through overlaying of the application code. The move to the cloud, together with more agile servicing and frequent updates, requires a less intrusive customization model, so that updates are less likely to affect custom solutions. This new model is called extensibility and has replaced customization through overlaying.

For more information, see Extensibility home page and the Develop and customize home page.

How do I prevent my models from being customized by customers or other partners?
You can block customizations of your model as described in Turn off model customization and deprecate functionality, or you can distribute deployable packages to your customers instead of distributing model files. See the section titled “How do I distribute my application to customers” later in this topic.

How can I define the scope of my models? How many models or packages should I create?
Designing models and model elements is no different than designing other types of software libraries. You should apply SOLID (object-oriented design) design principles. In addition, we recommend the following tips that are specific to the platform:

- If there are components in your solution that you want to ship and service more frequent than the rest, they are good candidates to place in a separate model and package.
- It is common practice to start with two packages (each with one model) at the initial stage of an implementation, one foundation package that contains extensions to the Microsoft platform packages and one application package that contains extensions to the Microsoft application packages. More models can be introduced on an as-needed basis.
- Existing packages can be subdivided into smaller packages when necessary. If your implementation is already live using one of your packages, avoid renaming a package, to help simplify lifecycle management.

Continuous delivery

Do I need build environments?
Yes, you should take advantage of the build and test automation tools provided in the build environments. You can deploy build environments from your Lifecycle Services (LCS) project. Creating daily builds and daily regression tests are key tools to enable the continuous delivery and maintain the quality of your application. Refer to Deploy and use an environment that supports continuous build and test automation for more details.

Do not use build environments for development activities. Do not keep a backup of your test database on these build VMs. Build VMs are designed to reset themselves to a known state with every build and whenever they are
updated with a Microsoft binary or platform updates from LCS. For example, if you apply a binary hotfix or platform update to a build VM, the VM prepares itself for the next build as part of the update. This will remove your customizations and also trigger a database synchronization.

**What strategy do I use for test automation?**

For test automation, concentrate on unit tests (use the SysTest framework) that are data independent or create their own data. Use a smaller number of functional scenario tests (based on Task Recorder) that rely on test data to execute. Scenario tests are more expensive to maintain. Unit tests can then be executed on any development environment easily and quickly. Review the Test Automation Pyramid blog article and refer to Automated testing guidance.

![Test Automation Pyramid](image)

Some key concepts to keep in mind:

- Write tests that run independently and do not assume any kind of ordering.
- Task recorder tests should be limited to functional scenarios tests.
- Write scenario tests after scenarios are complete and after completing unit tests.
- Create test helper classes when possible, so others on your team can leverage that as well.
- SysTest framework supports role-based testing, leverage this feature.

**How can I be more agile in my development?**

Deliver incremental features every sprint (2 weeks, preferred) or cycle (1 month). Maintain shippable quality of your application at the end of each sprint. Use Azure DevOps for work item tracking and always prioritize bugs over new features. A large bug backlog will quickly become a burden on your efficient delivery of new features and on the quality of your application.

**How do I manage test data?**

Create and manage your test database as follows:

- Start on a clean environment.
- Create all base data as required. Base data will serve as the starting point for all the tests.
- Take a backup (.bak) of your AxDB database.
- Share this backup with developers.

On a build environment, copy this backup over to the I:\DynamicsBackupDatabases (on some environment it may be a different drive than i). This database will be restored at the beginning of every build. This step is
How do I distribute my application to customers?

There are two artifacts you can use to distribute your application to your customers or partners: model files or deployable packages. Model files are design time artifacts that contain source code. Use model files if your customer is integrating your application with other third-party models or when you want to allow customization of your models. For more information, see Export and import models. Model files are the most common methods for ISVs to distribute solutions. Deployable packages are final applications. Use deployable packages with customers that will not be customizing or integrating your application with other third-party models. If you use deployable packages, your customer can only use or extend your application. They will not see or have access to your source code. To create a deployable package use the Visual Studio tools (Dynamics 365 > Deploy > Create Deployment package) or use a build environment. Build environments generate a deployable package with every successful build.

Development topologies

Should I develop on premises or in the cloud?

There are two modes of development: Cloud VMs and on-premises VMs available via a downloadable VHD. Use a combination of on premise VMs and cloud VMs for development.

- On premise dev VMs are cost effective if you already have the hardware, IT infrastructure, and Windows server licenses to support it.
- Use cloud VMs to scale out when projects require additional resources for a limited period of time. It is more cost effective than planning for worst-case capacity on premise.
- Connect all VMs (on premise and cloud VMs) to Azure DevOps for version control.

Use cloud VMs for build, functional testing, and demos. If you are running on your own Microsoft Azure subscription, turn them off when not in use.

Should I use a customer's dev environment?

If you are a partner, use your own VMs for development of your own intellectual property (IP), this is code and configuration data packages that are reusable across different customer implementations. For customer-specific implementations, you can use the customer’s dev VM. All customer subscriptions come with at least one development VM. Customers can pay for add-on dev VMs or run local dev VMs.

What are the benefits of MSDN subscriptions with respect to development?

The following is a summary of a Visual Studio (VS) with MSDN subscriptions benefits:

- Includes a Microsoft Azure subscription with a $50 monthly credit for Visual Studio Professional with MSDN and $150 for VS Enterprise with MSDN.
- Subscriptions come with lower dev/test rate, you will pay the Linux rates instead of the Windows rates.
- For more details, visit https://azure.microsoft.com/pricing/member-offers/msdn-benefits-details/

As a Microsoft partner, acquire Microsoft core competencies to earn free VS Enterprise with MSDN.
subscriptions. For example, an application development competency for a gold partner will earn 25 free MSDN Enterprise licenses in addition to the 10 licenses that come with the core benefits. For more details, visit Monthly Azure credit for Visual Studio subscribers. These benefits make cloud development very economical, for example:

- D12v2 VM list price = $470/month (4 core, 28 Gigs)
- D12v2 VM price if running on MSDN Azure subscription or any other dev/test subscription = $276/month
- Turn off 12 hours per day: 276/2 => $138/month
- Monthly credit (VS Professional with MSDN) => 138 - 50 = $88/month
- Monthly credit (VS Enterprise with MSDN) => 138 – 150 = Free

Here is another example:

- D13v2 machine list price = $843/month (8 core, 56 Gigs)
- D13v2 machine price if running on MSDN Azure subscription = $551/month
- Turn off 12 hours per day: 551/2 = $275.5/month
- Monthly credit (VS Professional with MSDN): 275.5 - 50 = $225.5/month
- Monthly credit (VS Enterprise with MSDN) => 275.5 – 150 = $125.5/month

Add an average of $15 monthly for storage (non premium) per VM.

Can more than one developer develop concurrently on the same VM?

This is not supported. However, you can provision more than one developer account on the same VM, they just cannot develop concurrently. For details, see Create new users on development machines.

If you are a Microsoft partner developing code for more than one customer, we recommend having at least one development VM per customer. You will need one additional VM for every additional developer working on a customer project. Development VMs can be thought of as disposable assets as long as your source code is checked into version control (Azure DevOps) and you keep a backup of test databases.

Customer implementation LCS projects

How many sandbox environments do I need within an LCS customer implementation project?

A customer subscription comes with two environments by default: a tier-2 sandbox environment and a production environment. You can use the tier-2 sandbox environment as a configuration and a UAT environment before the application goes live in production. After configuring the sandbox with the code and data that you need to go live (also known as your gold configuration), you can run your validation on the same environment. When your validation passes, restore your sandbox database to the point in time of its gold configuration. You can then deploy your code to production and copy the sandbox database to your production environment. You can also choose to have more than one sandbox environment that is tier-2 or higher, especially after your application is live. One sandbox can be used as a pre-production UAT environment, and the other sandboxes can be used for configuration, upgrade or other scenarios. You can purchase additional tier-2 or higher sandboxes.

The following servicing requests and tools are supported by LCS, which may help you decide whether one tier-2 sandbox is sufficient for your implementation.

1. Restore a sandbox database to a point in time.
2. Copy a sandbox database to a production environment (only allowed before the application is live in production).
3. Apply configuration data packages on a sandbox environment.
4. Apply configuration data packages on a production environment.
5. Refresh a sandbox database from production. Copy the production environment’s database to a tier-2 sandbox environment. This is typical after the application is live and you want to debug an issue or validate upcoming updates.
6. Apply updates (Hotfixes, customizations) to a sandbox environments for validation before applying them to a production environment.

For more information about planning an environment, see Environment planning.
In Platform update 4, the automated build process lets you prevent specific packages from being included in the deployable package in the build output. This capability can be important for customers that use automated testing. These customers might want to build and run their tests, but prevent them from being added to the deployable package that the build generates as output.

When customers that have an existing build definition from Platform update 3 or earlier upgrade, they won’t see the build definition automatically updated. To use the new feature, these customers must make a few manual edits to the build definition (see below for details).

The new feature exposes a new optional parameter for the package creation step in the build process. Because this parameter is managed by a build variable, you can easily adjust it.

1. In Microsoft Azure DevOps, on the Build & Release page, under Builds, on the All Definitions tab, find your build definition. Click the ellipsis (…), and then click Edit.

2. On the Variables tab, notice that the new build definition has a variable that is named **PackagingExclusions**.

3. In the **PackagingExclusions** variable, specify a comma-separated list of the names of packages that should not be packaged into the deployable package.

**NOTE**

The name of a package isn't necessarily the name of the model. Instead, the package name is typically the name of the folder where the model resides. Alternatively, you can copy and paste the package name from the descriptor file of one of the package's models. (In the XML, you can find the package name in the `ModelModule` field.)

For example, you have one package that is named MyCompanysAwesomeTests and another package that is named ContosoTaskRecordingTests, and you want to exclude both these packages from the deployable package. In this case, the value for the **PackagingExclusions** variable will look like this.
After you complete this setup, the build process will still build the code and run any tests that the packages contain. However, the deployable package that the build creates won't include those packages.

**Update an existing build definition after upgrade to Platform update 4 or later**

To use the new feature, you must manually update any existing build definitions that you deployed before Platform update 4.

NOTE
The feature can be added to a build definition only after you update the build virtual machine (VM) to Platform update 4 or later.

1. On the **Variables** tab, click **Add** at the bottom of the page.

2. In the **Name** column, enter **PackagingExclusions**. In the last column, select the **Settable at queue time** check box.

3. On the **Tasks** tab, find the **Generate Packages** task. Click to select it.

4. On the right side of the page, find the **Arguments** parameter. Click in the text box, and then press the End key or scroll to the end of the text box. The new build definition will have a new argument that passes the **PackagingExclusions** variable that you defined earlier. However, for an existing build definition, add a space and then the following text to the end of the parameter: `-ExclusionList "$(PackagingExclusions)"

The **Arguments** text box should now look like this.

5. Click **Save**.

You can now use the new feature as described.
Customers that work with solutions from third parties might receive different solution artifacts to use in their solution. Typically, these artifacts are distributed as code (in the form of models) or binaries (in the form of deployable packages). In some cases, third parties might provide some parts of their solution as code and other parts as a binary.

This topic outlines a recommended strategy for managing, distributing, and deploying these third-party solutions.

## Models from third parties

Any source code that is received from third parties must be compiled into a binary and included in a deployable package. Models should be installed on a development virtual machine (VM) and added to source control. From there, the build VM can pick up the source code, build it, and include it in a deployable package. Other developers can just synchronize the model from Microsoft Azure DevOps to their development VMs. They don’t have to manually install it.

For information about how to install a model on a development VM, see Export and import models.

After you install the model, follow these steps to add the new model to source control.

2. Open Application Explorer by clicking View > Application Explorer.
3. Right-click the AOT root node, and then click Model view.
4. In the list of models, find the new model that you installed. Make a note of the name of the package that contains the model. The package name appears in parentheses after the model name. For example, in the following illustration, the Tax Books, Tax Engine Configuration, and Tax Engine Interface models all belong to the package that is named TaxEngine.

![Application Explorer](image)

5. Open Source Control Explorer by clicking View > Other Windows > Source Control Explorer.
6. Navigate to the metadata folder that is mapped on this development VM, such as MyProject/Trunk/Main/Metadata.
7. In the metadata folder, find the folder for the package that contains the new model. Right-click the package folder, and then click Add Items to Folder.
8. In the Add to Source Control dialog box, select the Descriptor folder and the folder that has the name of the model. Some models may also contain referenced DLLs in the bin folder. If these exist you’ll need to also include the appropriate DLL files from the bin folder. Once all files have been selected, click Next.
After checking in the package binaries into source control as described in the previous section, you include the binaries in deployable packages generated during build automation. There are two options:

- On a build virtual machine, the standard legacy pipeline will automatically find and include the binaries into the deployable package it generates.
- Using the new pipeline or using the new packaging task in the legacy pipeline, review the create deployable packages documentation. The documentation has examples for including source-controlled binaries in deployable packages.

Deploying third-party code
Because the models and runtime packages are in source control, other developers who use other development environments can just synchronize the models and packages to their workspace by using the **Get latest** feature of source control.

As of Platform update 4, the automated build process will also pick up the runtime packages. Therefore, dependencies in packages that are built will be resolved correctly. This feature is also available for Platform update 3 and Platform update 2 through a hotfix.

In Platform update 6, the build process will include this runtime package in the final deployable package. This allows customers to take the deployable package from the build and have one package to deploy to their environments. The one package includes both custom solutions and all the third party solutions.
Customers can update the software in their environments by applying software deployable packages. These packages can originate from the customers themselves in the form of customizations. They can also be provided by partners and independent software vendors (ISVs). Microsoft recommends that customers combine all these various packages into a single package before they apply them to an environment. For customers who have self-service environments, this approach is a hard requirement.

This topic outlines the best practices for creating and managing an all-in-one deployable package.

**IMPORTANT**

- The enforcement of all-in-one packages will be done in phases. Requests to extend the support for deployable packages that are not all-in-one deployable packages ended on October 31, 2020.
- If a payment connector is currently deployed in your environment, you will have to create a payment connector package and include it in the all-in-one deployable package.
- If you currently use Microsoft Dynamics 365 Commerce functionality for the retail point of sale, you will also have to synchronize self-service installers. For Commerce Scale Unit, please see Deploy the package to CSU.

**What is an all-in-one deployable package?**

An all-in-one deployable is a software deployable package that contains all the models and binaries that you currently have in an environment. Think of it as a single package that represents all the non-Microsoft software in an environment.

For example, you have two environments: **SandboxTest** and **SandboxPreProd**.

If your software deployable package contains CustomizationA, CustomizationB, and ISV1, it's a fully deployable package for the SandboxTest environment. This is because it exactly matches the model list. It's also a fully deployable package for the SandboxPreProd environment because it has all the models that are installed, plus CustomizationB.

However, if your software deployable package contains only CustomizationA, it isn't fully deployable for either environment. The package is missing some of the models that are already installed.

**How do I create an all-in-one deployable package?**

There are two primary methods for creating an all-in-one deployable package:

- If you're using the continuous integration/continuous deployment model, you're already creating all-in-one deployable packages.
- If you don’t have a build environment, you can create a package in Microsoft Visual Studio. For more
What if my ISV packages don't contain source code?

ISVs can choose whether to share their source code with you. If they don't share it, they will provide a binary-only package. This package can easily be managed into an all-in-one deployable package. For instructions, see Manage third-party models and runtime packages by using source control.

How can I deploy ISV licenses?

ISVs can send license deployable packages to provide or update a license. However, for self-service environments licenses should also be included in an all-in-one deployable package. You can add a task to your build or release pipelines to add any licenses you have to a deployable package. For more information, see Add license files to a deployable package in Azure Pipelines.

Why are these packages important?

The best practice of using fully deployable packages helps reduce the complexity and number of packages that are applied to a given environment. In some circumstances, installation of different packages can change the behavior of your environment. For example, if you install ModelA and then ModelB instead of ModelB and then ModelA.

In addition, this approach is a hard requirement for self-service environments. This is because those environments use containerization technology and build a brand-new environment every time that you apply a package. If you apply ModelA today and then apply only ModelB tomorrow, you will effectively uninstall ModelA.
In Platform update 6, a new task in the automated build definition updates the models in the source package and deployable package of the build output with the version of the build that produced them.

Build definitions that were created before Platform update 6 must be manually updated to include this task. See the Updating an existing build definition section later in this topic.

Version numbers

Even though models are compiled into one package, the metadata information of all models is retained inside the binary package. This information can be reviewed from Microsoft Dynamics Lifecycle Services (LCS) or from the client.

In LCS, follow these steps to find the version numbers of models that are installed in an environment.

1. On the Full details page for the environment, under Environment Version Information, click the View detailed version information link.
2. On the Installed updates page, in the Machine name field, select an Application Object Server (AOS) computer.
3. In the table list, find the Publisher name field of the model, and expand the list by clicking the arrow icon. A full list of all models from that publisher is shown. The version number is shown in the Version column.

In the client, follow these steps to find the version numbers of models that are installed in an environment.

1. Open the URL for the environment, and sign in.
2. After the dashboard is loaded, click the gear symbol at the upper right of the page, and then click About.
3. In the dialog box that appears, expand Loaded Packages and their Models. Find the package where the model resides, and expand the list by clicking the arrow icon. The list of models for the package is shown, together with the version number.

All version numbers are in .NET assembly format. They consist of four numbers that are separated by a dot (.), such as 1.2.3.4.

The purpose of model versioning

As code is updated, the build is used to produce new packages that can be deployed to environments. Microsoft Azure DevOps tracks the changes that have been included in each build since the previous build. When the version number of the build is included in the models that are produced, it provides end-to-end traceability of the code changes that are available in a specific environment. You can find the build number and then review the changes that are included in that build in Azure DevOps. For customers and partners that use builds on different branches, or that use different build definitions for nightly builds, gated check-in builds, or deployment builds, each build can have a different versioning scheme. This approach helps differentiate the model metadata in the deployable packages and tie them back to their originating build definition.

Setting up versioning

For build definitions that are created by Platform update 6 or newer deployments, the task to include build version in models is automatically added and active. The default build number of a new build definition in Azure DevOps consists of the year, month, and day, and the incremental number of the build for that day. For more
information about build numbers in Azure DevOps, and the options that are available, see Build definition options on the Microsoft Visual Studio docs site.

The automated build will apply the build version number to the models that are built.

Preventing models from being updated

By default, the build task assigns versions only to models that are in layers above the ISP layer. Therefore, customers can consume code models from third-party vendors without overwriting the version numbers that are supplied in their models. However, you can also prevent other models from having their version numbers overwritten during the build, regardless of layer. When you edit the build definition, on the Variables tab, in the ModelVersionExclusions variable, supply a comma-separated list of model names to exclude.

Updating models in lower layers

For third parties that develop solutions in the ISV or ISP layer, a manual change must be made to the build definition to automatically set model versions in those layers.

1. Edit the build definition. On the Tasks tab, click the Set Model Versions task.
2. In the Arguments field, add the following option at the end of the existing list of arguments:
   - UpdateLayersAbove 7

Updating an existing build definition

For build definitions that were created before Platform update 6, a new task must be manually added to the build definition.

   NOTE
   This feature can be added to a build definition only after the build virtual machine (VM) has been updated to Platform update 6 or later.

1. In Azure DevOps, on the Build & Release page, under Builds, on the All Definitions tab, find your build definition.
2. Click the ellipsis (…), and then click Edit.
3. On the Tasks tab, click + Add Task at the bottom of the page.
4. In the Add tasks pane on the right side of the page, on the Utility tab, scroll down to find the PowerShell task.
5. Hover the mouse pointer over the task, and click the Add button that appears.
6. In the list of tasks on the left side of the page, click to select the **PowerShell Script** task that is added.

7. On the right side of the page, change the **Display name**, **Script Path**, and **Arguments** properties to reflect the required settings.

8. In the list of tasks on the left side of the page, drag the **Set Model Versions** task so that it's between the **Prepare for build** and **Build the solution** tasks.

9. On the **Variables** tab, click + **Add** at the bottom of the list of variables. In the first column, for the **Name** variable, enter **ModelVersionExclusions**.

10. Click **Save** to save the new task.
This topic is intended to help architects and developers make sound design decisions when they implement integration scenarios.

The topic describes integration patterns, integration scenarios, and integration solutions and best practices. However, it doesn't include technical details about how to use or set up every integration pattern. It also doesn't include sample integration code.

### NOTE
When providing guidance and discussing scenarios for choosing a pattern, data volume numbers are mentioned. These numbers must be used only to gauge the pattern and must not be considered as hard system limits. The absolute numbers will vary in real production environments due to various factors, configurations are only one aspect of this scenario.

The following table lists the integration patterns that are available.

<table>
<thead>
<tr>
<th>PATTERN</th>
<th>DOCUMENTATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Platform integration</td>
<td>Microsoft Power Platform integration with Finance and Operations apps</td>
</tr>
<tr>
<td>OData</td>
<td>Open Data Protocol (OData)</td>
</tr>
<tr>
<td>Batch data API</td>
<td>Recurring integrations Data management package REST API</td>
</tr>
<tr>
<td>Custom service</td>
<td>Custom service development</td>
</tr>
<tr>
<td>Consume external web services</td>
<td>Consume external web services</td>
</tr>
<tr>
<td>Excel integration</td>
<td>Office integration overview</td>
</tr>
</tbody>
</table>

### NOTE
For on premise deployments, the only supported API is the Data management package REST API. This is currently available on 7.2, platform update 12 build 7.0.4709.41184.

**Synchronous vs. asynchronous integration patterns**

Processing can be either synchronous or asynchronous. Often, the type of processing that you must use determines the integration pattern that you choose.

A *synchronous* pattern is a blocking request and response pattern, where the caller is blocked until the callee has finished running and gives a response. An *asynchronous* pattern is a non-blocking pattern, where the caller submits the request and then continues without waiting for a response.
The following table lists the inbound integration patterns that are available.

<table>
<thead>
<tr>
<th>PATTERN</th>
<th>TIMING</th>
<th>BATCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>OData</td>
<td>Synchronous</td>
<td>No</td>
</tr>
<tr>
<td>Batch data API</td>
<td>Asynchronous</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Before you compare synchronous and asynchronous patterns, you should be aware that all the REST and SOAP integration application programming interfaces (APIs) can be invoked either synchronously or asynchronously.

The following examples illustrate this point. You can’t assume that the caller will be blocked when the Open Data Protocol (OData) is used for integration. The caller might not be blocked, depending on how a call is made.

<table>
<thead>
<tr>
<th>PATTERN</th>
<th>SYNCHRONOUS PROGRAMMING PARADIGM</th>
<th>ASYNCHRONOUS PROGRAMMING PARADIGM</th>
</tr>
</thead>
<tbody>
<tr>
<td>OData</td>
<td>DbResourceContextSaveChanges</td>
<td>DbResourceContextSaveChangesAsync</td>
</tr>
<tr>
<td>Custom service</td>
<td>httpRequestGetResponse</td>
<td>httpRequestBeginGetResponse</td>
</tr>
<tr>
<td>SOAP</td>
<td>UserSessionServiceGetUserSessionInfo</td>
<td>UserSessionServiceGetUserSessionInfo Async</td>
</tr>
<tr>
<td>Batch data API</td>
<td>ImportFromPackage</td>
<td>BeginInvoke</td>
</tr>
</tbody>
</table>

Both OData and custom services are synchronous integration patterns, because when these APIs are called, business logic is immediately run. Here are some examples:

- If OData is used to insert product records, the records are immediately inserted as part of the OData call.
- If a custom service is used to look up on-hand inventory, business logic is immediately run as part of the JSON/SOAP call, and an inventory sum number is immediately returned.

Batch data APIs are considered asynchronous integration patterns, because when these APIs are called, data is imported or exported in batch mode. For example, calls to the ImportFromPackage API can be synchronous. However, the API schedules a batch job to import only a specific data package. The scheduling job is quickly returned, and the work is done later in a batch job. Therefore, batch data APIs are categorized as asynchronous.

Batch data APIs are designed to handle large-volume data imports and exports. It’s difficult to define what exactly qualifies as a large volume. The answer depends on the entity, and on the amount of business logic that is run during import or export. However, here is a rule of thumb: If the volume is more than a few hundred thousand records, you should use the batch data API for integrations.

In general, when you’re trying to choose an integration pattern, we recommend that you consider the following questions:

- Is there a business requirement that the integration should be in real time?
- What is the requirement for the peak data volume?
- What is the frequency?

**Error handling**

When you use a synchronous pattern, success or failure responses are returned to the caller. For example, when an OData call is used to insert sales orders, if a sales order line has a bad reference to a product that doesn’t exist, the response that the caller receives contains an error message. The caller is responsible for handling any errors in the response.
When you use an asynchronous pattern, the caller receives an immediate response that indicates whether the scheduling call was successful. The caller is responsible for handling any errors in the response. After scheduling is done, the status of the data import or export isn’t pushed to the caller. The caller must poll for the result of the corresponding import or export process, and must handle any errors accordingly.

Typical scenarios and patterns that use OData integrations

Here are some typical scenarios that use OData integrations.

**NOTE**

Use of OData for Power BI reports is discouraged. Using entity store for such scenarios is encouraged.

Create and update product information

A manufacturer defines and configures its product by using a third-party application that is hosted on-premises. This manufacturer wants to move its production information from the on-premises application to Finance and Operations. When a product is defined, or when it’s changed in the on-premises application, the user should see the same change, in real time.

<table>
<thead>
<tr>
<th>DECISION</th>
<th>INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is real-time data required?</td>
<td>Yes</td>
</tr>
<tr>
<td>Peak data volume</td>
<td>1,000 records per hour*</td>
</tr>
<tr>
<td>Frequency</td>
<td>Ad hoc</td>
</tr>
</tbody>
</table>

Occasionally, many new or modified production configurations will occur in a short time.

**Recommended solution**

This scenario is best implemented by using the OData service endpoints to create and update product information in Finance and Operations.

In Finance and Operations:

- Determine all the entities that are required for the integration.
- Make sure that the OData service endpoints are available for the same set of entities.

In the third-party application:

- When product information is created or modified in the third-party application, an OData call is made to Finance and Operations to make the same change.

Read the status of customer orders

A company has a self-hosted customer portal where customers can check the status of their orders. Order status information is maintained in the application.

<table>
<thead>
<tr>
<th>DECISION</th>
<th>INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is real-time data required?</td>
<td>Yes</td>
</tr>
<tr>
<td>Peak data volume</td>
<td>5,000 records per hour</td>
</tr>
<tr>
<td>Frequency</td>
<td>Ad hoc</td>
</tr>
</tbody>
</table>
Recommended solution
This scenario is best implemented by using the OData service endpoints to read order status information.

<table>
<thead>
<tr>
<th>DECISION</th>
<th>INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is real-time data required?</td>
<td>Yes</td>
</tr>
<tr>
<td>Peak data volume</td>
<td>1,000 records per hour</td>
</tr>
<tr>
<td>Frequency</td>
<td>Ad hoc</td>
</tr>
</tbody>
</table>

Approve BOMs
A company uses a product lifecycle management (PLM) system that is hosted on-premises. The PLM system has a workflow that sends the finished bill of materials (BOM) information to the application for approval.

<table>
<thead>
<tr>
<th>DECISION</th>
<th>INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is real-time data required?</td>
<td>Yes</td>
</tr>
<tr>
<td>Peak data volume</td>
<td>1,000 records per hour</td>
</tr>
<tr>
<td>Frequency</td>
<td>Ad hoc</td>
</tr>
</tbody>
</table>

Recommended solution
This scenario can be implemented by using an OData action.

<table>
<thead>
<tr>
<th>DECISION</th>
<th>INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is real-time data required?</td>
<td>Yes</td>
</tr>
<tr>
<td>Peak data volume</td>
<td>1,000 records per hour</td>
</tr>
<tr>
<td>Frequency</td>
<td>Ad hoc</td>
</tr>
</tbody>
</table>

In Finance and Operations:

- Determine the entity that is required in order to read order status information.
- Make sure that the OData service endpoint is available for the entity.

On the customer portal site:

- When a customer checks the status of an order, make a real-time OData call to Finance and Operations to read the corresponding order and retrieve its status.

You can find an example of this type of OData action in BOMBillOfMaterialsHeaderEntity::approve.

Typical scenarios and patterns that use a custom service

Here are some typical scenarios that use a custom service.

Look up on-hand inventory
An energy company has field workers who schedule installation jobs for heaters. This company uses the application for the back office and third-party software as a service (SaaS) to schedule appointments. When field workers schedule appointments, they must look up inventory availability to make sure that installation parts are available for the job.

<table>
<thead>
<tr>
<th>DECISION</th>
<th>INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is real-time data required?</td>
<td>Yes</td>
</tr>
</tbody>
</table>
**Recommended solution**
This scenario can be implemented by using a custom service.

In Finance and Operations:

- Create a custom service to calculate the physical on-hand inventory for a given item.

In the scheduling application:

- Make a real-time call to a custom service endpoint, through either SOAP or REST, to retrieve inventory information for the selected item.

**NOTE**
You can find an example of this type of custom service in the Retail Real Time Services implementation: `RetailTransactionServiceInventory::inventoryLookup`.

You can also use the `inventorySiteOnHand` entity to achieve the same result. Sometimes, you can use multiple methods to expose the same data and business logic, and all the methods are equally valid and effective. In this case, choose the method that works best for a given scenario and that a developer is most comfortable with.

**Typical scenarios and patterns that use batch data integrations**

Here are some typical scenarios that use batch data APIs.

**Import large volumes of sales orders**
A company receives a large volume of sales orders from a front-end system that runs on-premises. These orders must periodically be sent to the application for processing and management.

<table>
<thead>
<tr>
<th>DECISION</th>
<th>INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is real-time data required?</td>
<td>No</td>
</tr>
<tr>
<td>Peak data volume</td>
<td>200,000 records per hour</td>
</tr>
<tr>
<td>Frequency</td>
<td>One time every five minutes</td>
</tr>
</tbody>
</table>

**Recommended solution**
This scenario is best implemented by using batch data APIs.

In Finance and Operations:

- Determine all the entities that are required for the integration.
- Make sure that data management is enabled for the entities.

In the on-premises system:

- Use the REST batch data API to import files.

**Export large volumes of purchase orders**
A company generates a large volume of purchase orders in Finance and Operations and uses an on-premises inventory management system to receive products. Purchase orders must be moved from Finance and Operations to the on-premises inventory system.

<table>
<thead>
<tr>
<th>DECISION</th>
<th>INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is real-time data required?</td>
<td>No</td>
</tr>
<tr>
<td>Peak data volume</td>
<td>300,000 records per hour</td>
</tr>
<tr>
<td>Frequency</td>
<td>One time per hour</td>
</tr>
</tbody>
</table>

**Recommended solution**

This scenario is best implemented by using batch data APIs.

In Finance and Operations:

- Determine all the entities that are required for the integration.
- Make sure that data management is enabled for the entities.
- If incremental push is required, make sure that change tracking can be enabled on the entities.

In the on-premises inventory system:

- Use the REST batch data API to export the file from Finance and Operations and import it into the inventory system.

**Typical scenarios and patterns that call external web services**

It's typical that the application calls out to an external web service that is hosted either on-premises or by another SaaS provider. In this case, the application acts as the integration client. When you write an integration client, you should follow the same set of best practices and guidelines that you follow when you write an integration client for any other application. For a simple example, see Consuming external web services.

**IMPORTANT**

Because of security requirements, production and sandbox environments support only secured communication that uses Transport Layer Security (TLS) 1.2 or later. In other words, the target web service endpoint that the application calls out to must support TLS 1.2 or later. If the target service endpoint doesn't meet this requirement, calls fail. The exception error message resembles the following message:

*Unable to read data from the transport connection: An existing connection was forcibly closed by the remote host.*

If you can't modify the target service so that it uses TLS 1.2 or later, you can work around this issue by introducing a broker service and making a two-hop call, as shown in the following illustration.
Priority-based throttling is enabled by default starting in Dynamics 365 Finance version 10.0.19. To learn about how you can prepare your environment prior to updating to version 10.0.19, see FAQ document. To test this capability, configure the integration priorities on the Throttling priority mapping page.

KB 4615823 includes two important fixes that directly impact the priority-based throttling experience.

This fix ensures that 429 HTTP messages are sent when a throttling event occurs in your environment and that the events reflect the correct threshold calculations for your environment.

The fix is available as part of the quality update for 10.0.17. We recommend you install this version to ensure that your environment is ready for priority-based throttling.

Priority-based throttling introduces service protection settings that prevent the over-utilization of resources to preserve the system’s responsiveness and ensure consistent availability and performance for environments running Finance and Operations apps.

You have the ability to set relative priority for the OData and custom service-based integrations, depending on your business-critical need for these integrations. The throttling manager will then honor these priorities set for these requests.

- For OData and custom service requests, an error which states "Too many requests" will be sent when system health and performance are affected.
- You can query throttling events on the Lifecycle Services Monitoring page.

The Throttling Priority Mapping page is used to assign priorities for integrations so that priorities can be honored when requests are throttled.

Setting appropriate priorities ensures that low-priority integrations will be throttled before high-priority integrations. For more information about how to set up integration, see Enable connectivity with external services.

There are two kinds of applications are supported in Microsoft Azure Active Directory (Azure AD):

- User based - This flow uses a username and password for authentication and authorization.
- Azure AD app based - A confidential client is an application that can keep a client password confidential. The authorization server assigned this client password to the client application.

For more information, see Authentication.

Configure priorities for integrations

After you have registered your service in Azure AD and in your Finance and Operations apps, you can set up priorities for integrations.
NOTE
You must be assigned the System administrator or Integration priority manager role to complete the set up.

1. In Finance and Operations apps, go to System administration > Setup > Throttling priority mapping.
2. Select New.
3. In the Authentication type field, select User or Azure AD application based on your integration scenario.
4. If Azure AD application type is selected, in the Client ID field select the application that you registered in the Azure Active Directory application.
5. If User type is selected, in the User ID field select an appropriate service account user ID.
6. Assign the appropriate priority and then select Save.

Retry operations

When a request is throttled, the system provides a value indicating the duration before any new requests from the user can be processed. When a request is throttled and a 429 error occurs, the response header will include a Retry-After interval, which can be used to retry the request after a specific number of seconds. The following example shows this operation.

```csharp
if (!response.IsSuccessStatusCode)
{
    if ((int)response.StatusCode == 429)
    {
        int seconds = 30;
        // Try to use the Retry-After header value if it is returned.
        if (response.Headers.Contains("Retry-After"))
        {
            seconds = int.Parse(response.Headers.GetValues("Retry-After").FirstOrDefault());
        }
        Thread.Sleep(TimeSpan.FromSeconds(seconds));
    }
    // Retry sending the request.
}
```

Monitoring

To have a successful onboarding experience with the throttling capability, you must also be able to monitor your OData and custom service integration patterns. Microsoft Dynamics Lifecycle Services (LCS), which is the administration center, contains a collection of monitoring and diagnostics tools that can help ensure that you have an accurate view of the environments you manage. For more information, see Monitoring and diagnostics tools in Lifecycle Services (LCS).

You can use a set of predefined queries to get raw logs for an issue. You can then export the logs for a more advanced analysis. The following types of queries are available:

- All throttling events
- Requests throttled

Access the Monitoring and diagnostics portal

1. In LCS, open the appropriate project.
2. In the Environments section, select the environment to view, and then select Full details.
3. On the **Environment details** page, select **Environment monitoring** to open the Monitoring and diagnostics portal.

4. On the **Environment monitoring** page, select the **Activity** tab to view the **Raw logs** page.

5. Select the **Query name**, and then select **Requests throttled** for all OData and custom services requests that have been throttled.

**NOTE**

In LCS, the **All throttling events** report has been removed. The report was introduced prior to priority-based throttling becoming generally available as a way to highlight sample requests that would be subjected to throttling. The **Requests throttled** report offers a more consistent view of requests subjected to throttling and should be used to gain insights into your environment.
This topic provides answers to some frequently asked questions about priority-based throttling for Open Data Protocol (OData) and custom service-based integrations.

How do I access the Data management Yammer group?
To access the Data management Yammer group, go to Data management Yammer group.

When will throttling be enabled by default?
The throttling feature has been in preview since Finance 10.0.13 and will be enabled by default starting in version 10.0.19. You can manually trigger throttling in your environment by setting the priority for your OData and Custom service requests. However, starting with the release of Finance 10.0.19, throttling will be automatically enabled even if you haven’t set priorities.

Finance version 10.0.19 will be available for pre-release in late April 2021 and will be generally available in June 2021.

Will a retry request receive preferential treatment over a new request?
No.

Will my environment be subject to any API limits?
No. At this time, environments are not subject to any API limits.

Telemetry data is collected to assess the impact of the workloads across environments. This data helps establish baseline API limits as we define a roadmap for introducing usage-based limits.

Is there a report that determines when throttling might occur?
Yes. A report is available and can be accessed through the Raw logs on the Environment monitoring page in LCS. The requests that are listed in this view are likely to be throttled when this feature is turned on by default starting in Finance version 10.0.19.

Will throttling affect the Data Import/Export Framework (DIXF) and batch?
No. Throttling is only for OData and custom service integrations.

Is throttling available for on-premises environments?
No. Throttling is not available for on-premises environments.

In Preview, will my requests be throttled if priorities aren’t configured?
No, because only the telemetry is available. The actual throttling occurs if you configure priorities. We
Can throttling be enabled on dev boxes?
No. You can only set priority and trigger throttling in sandbox or production environments.

What happens to requests if the user didn't retry a throttled request?
Currently, if a request isn't retried when a 429 error is received, the request won't be processed.

Will historic throttling information be used to advise me when I resize environments?
Yes. For one month, you can export the information to Excel for more analysis and archiving.

Is throttling functionality version-specific? If it is, which version is it available in?
Priority-based throttling will be available in Preview starting with the version 10.0.13 of Finance and Operations apps.

Are there plans to provide an option for the Priority mapping grid entry?
Microsoft will consider this request in a future release.

Will the requests of my interactive users be throttled?
No. There will be no impact on the requests of interactive (online) users.

If I experience a performance issue in Dynamics 365 Finance while a page is being loaded or a business document is being processed, how does that performance issue differ from throttling?
Throttling helps maintain a healthy system when there is a resource constraint. It won't affect any page actions.

How can I determine the wait time before I retry a throttled request?
When a request is throttled, the response header includes a time that will be used in retry logic. You can use the Retry-After HTTP header to fetch the value that will be provided in seconds. For example, Retry-After: 60.

What is the message that is shown as part of the 429 HTTP response?
You will receive the message, "This request could not be processed at this time due to system experiencing high resource utilization. Retry the request after (0) seconds". Where [0] will have the dynamically calculated retry-after time interval.

Does throttling apply to Microsoft services?
Starting in Finance version 10.0.19, the following Microsoft services are initially exempt, and throttling doesn’t apply to them:
- Document Routing Agent (DRA)
Is it recommended to use a dedicated integration account instead of just the generic admin user account?

Yes, we strongly recommend this approach. As these service protection settings are set up for user-specific values, using the same user account for most or all of your integration will limit your ability to assign relative priorities across your integration needs. Further, if the same user account is used, all integration requests that originate from that user account will be subjected to throttling.

Does throttling depend on the tier that my environment is running on?

In the initial release, no. Throttling will calculate its threshold based on the resources that are available for each environment.

Is throttling functionality legal entity–specific?

No.

Does throttling affect bring your own database (BYOD) export?

No.

Will throttling monitoring be available in Application Insights?

Monitoring will be onboarded to any available tool in the future.

If a production environment regularly runs out of resources, will Microsoft have to resize it?

Yes. Sizing estimate will also have to be revalidated and uploaded.

After April 2021, will the system be able to override priority-based throttling?

Though these services are exempt, telemetry is being collected on the performance and impact of these services on the overall system health.

The owners of the exempt services are prioritizing the implementation of 429 handlers by the end of 2021. At that time, the services will no longer be exempt and throttling will apply. Notification will be provided ahead of these changes, and the documentation will be updated.

Even if these services implement their own handlers, we still recommended that you have client-side handling. Consider implementing the 429 handler with **retry-after** logic.
The system will use default values if no priorities are configured after April 2021.

Can the throttling engine be configured (thresholds)?

No.
This topic describes the Data management framework’s package representational state transfer (REST) application programming interface (API). The package API lets you integrate by using data packages. The REST API can be used with both cloud deployments and on-premises deployments.

Although on-premises support has been added, API names haven’t been changed. Therefore, Microsoft can keep a single API set for both cloud deployments and on-premises deployments.

### Choosing an integration API

Two APIs support file-based integration scenarios: the Data management framework’s package API and the recurring integrations API. Both APIs support both data import scenarios and data export scenarios. The following table describes the main decision points that you should consider when you’re trying to decide which API to use.

<table>
<thead>
<tr>
<th>Decision Point</th>
<th>Recurring Integrations API</th>
<th>Data Management Framework’s Package API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduling</td>
<td>Scheduling in Finance and Operations apps</td>
<td>Scheduling outside Finance and Operations apps</td>
</tr>
<tr>
<td>Format</td>
<td>Files and data packages</td>
<td>Only data packages</td>
</tr>
<tr>
<td>Transformation</td>
<td>Support for Extensible Stylesheet Language Transformations (XSLT) if the data file is in XML format</td>
<td>Transformations that are external to the system</td>
</tr>
<tr>
<td>Supported protocols</td>
<td>SOAP and REST</td>
<td>REST</td>
</tr>
<tr>
<td>Service type</td>
<td>Custom service</td>
<td>Open Data Protocol (OData) action</td>
</tr>
</tbody>
</table>

If you decide that the recurring integrations API meets your requirement better than the Data management framework’s package API, see [Recurring integrations](#). The rest of this topic discusses the Data management framework’s package API.

### Authorization

The Data management framework’s package API uses OAuth 2.0 to authorize access. The API must be called by using a valid OAuth access token. For more information about OAuth 2.0 and Microsoft Azure Active Directory (Azure AD), see [Authorize access to web applications using OAuth 2.0 and Azure Active Directory](#). For on-premises deployments, Active Directory Federation Services (AD FS) is used for authorization.
NOTE
When you use the Client Credentials Grant flow, the application maintains an access control list. You can find the access control list at System administration > Setup > Azure Active Directory applications. The Azure Active Directory applications page shows the approved client IDs and the user security mapping that should be enforced when the API is called by using the Client Credentials Grant flow.

For on-premises deployments, this list must have a valid client ID from AD FS. Additionally, for on-premises use, `<baseurl>` in the following examples must append /namespaces/AXSF when a connection is made.

Import APIs
The following APIs are used to import files (data packages).

GetImportStagingErrorFileUrl
The GetImportStagingErrorFileUrl API is used to get the URL of the error file containing the input records that failed at the source to the staging step of import for a single entity. An empty string is returned if no error file is generated.

POST /data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.GetImportStagingErrorFileUrl

Body
{
  "executionId": "<string>",
  "entityName": "<string>"
}

Successful Response:

HTTP/1.1 200 OK
{
  "@odata.context": "https://<baseurl>/data/$metadata#Edm.String",
  "value": "<errorfileurl>"
}

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>string executionId</td>
<td>Execution ID of import. This is called as Job ID in the UI.</td>
</tr>
<tr>
<td>string entityName</td>
<td>Name of the entity for which to get the error file.</td>
</tr>
</tbody>
</table>

GenerateImportTargetErrorKeysFile
The GenerateImportTargetErrorKeysFile API is used to generate an error file containing the keys of the import records that failed at the staging to target step of import for a single entity.

If this API returns true, then use the GetImportTargetErrorKeysFileUrl API to get the URL of the generated error keys file.
The **GetImportTargetErrorKeysFileUrl** API is used to get the URL of the error file that contains the keys of the import records that failed at the staging-to-target step of import for a single entity.

If the error file is available, this API returns the URL. If the error file is still being generated, or if there is no error file, the API returns an empty string.

### IMPORTANT
Before you call this API, call the **GenerateImportTargetErrorKeysFile** API to generate the error file. If the **GenerateImportTargetErrorKeysFile** API returns **true**, call this API in a loop until it returns a non-empty string. If the **GenerateImportTargetErrorKeysFile** API returns **false**, this API will always return an empty string, because there are no errors.

**Pseudocode example**
errorsExist = GenerateImportTargetErrorKeysFile(executionId, entityName)

if (errorsExist)
{
    errorFileUrl = null

    while (errorFileUrl is not a non-empty string)
    {
        errorFileUrl = GetImportTargetErrorKeysFileUrl(executionId, entityName)
        if (errorFileUrl is not a non-empty string)
        {
            wait for some time before retrying
        }
    }
}

POST
/data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.GetImportTargetErrorKeysFileUrl

Body
{
    "executionId":"<string>",
    "entityName":"<string>"
}

Here is an example of a successful response.

HTTP/1.1 200 OK
{
    "@odata.context":"https://<baseurl>/data/$metadata#Edm.String",
    "value":"<errorkeysfileurl>"
}

Input parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>string execId</td>
<td>The execution ID of the import. This is called as Job ID in the UI.</td>
</tr>
<tr>
<td>string entityName</td>
<td>The name of the entity to get the error file for.</td>
</tr>
</tbody>
</table>

Output parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>string errorkeysfileurl</td>
<td>The URL of the error keys file, if the file is available. If the error file is still being generated, or if no errors exist, the method returns an empty string.</td>
</tr>
</tbody>
</table>

**GetAzureWritableUrl**

The **GetAzureWritableUrl** API is used to get a writable blob URL. The method includes a shared access signature (SAS) token that is embedded in the URL. You can use this method to upload a data package to the Azure Blob storage container. For on-premises deployments, this API will still return the URL that has been abstracted to local storage.
POST /data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.GetAzureWriteUrl
BODY
{
    "uniqueFileName": "<string>
}

Here is an example of a successful response.

HTTP/1.1 200 OK
{
    "@odata.context": "https://<baseurl>/data/$metadata#Edm.String",
    "value": "{\"BlobId\":\"<GUID>\",
    \"BlobUrl\": \"https://<baseurl_id>.blob.core.windows.net/dmf/<uniqueFileName>?<SAS Token>\"}"
}

### Input parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>string uniqueFileName</td>
<td>A unique file name that is used to track blob IDs. You can include a globally unique identifier (GUID) to help guarantee a unique file name.</td>
</tr>
</tbody>
</table>

### Output parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>string BlobId</td>
<td>The blob ID of the allocated blob container.</td>
</tr>
<tr>
<td>string BlobUrl</td>
<td>A URL that has an embedded SAS token. The URL can be used to write to Blob storage.</td>
</tr>
</tbody>
</table>

### ImportFromPackage

The **ImportFromPackage** API is used to initiate an import from the data package that is uploaded to the Blob storage that is associated with your implementation. For on-premises deployments, the import will be initiated from the local storage that the file was uploaded previously to.

**NOTE**

The **ImportFromPackage** API supports composite entities. However, the limitation is that there can be only one composite entity in a package.
POST /data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.ImportFromPackage
BODY
{
  "packageUrl":"<string>",
  "definitionGroupId":"<string>",
  "executionId":"<string>",
  "execute":<bool>,
  "overwrite":<bool>,
  "legalEntityId":"<string>
}

Here is an example of a successful response.

HTTP/1.1 200 OK
{
  "@odata.context":"https://<baseurl>/data/$metadata#Edm.String",
  "value":"<executionId>"
}

Input parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>string packageUrl</td>
<td>The URL of the data package in the Blob storage that is associated with a Finance and Operations app.</td>
</tr>
<tr>
<td>string definitionGroupId</td>
<td>The name of the data project for import.</td>
</tr>
<tr>
<td>string executionId</td>
<td>The ID to use for the job. This is called as Job ID in the UI. If an empty ID is assigned, a new execution ID will be created.</td>
</tr>
<tr>
<td>bool execute</td>
<td>Set this parameter to True to run the target step. Otherwise, set it to False.</td>
</tr>
<tr>
<td>bool overwrite</td>
<td>This parameter must always be set to False when a composite entity is used in a package. Otherwise, set it to True.</td>
</tr>
<tr>
<td>string legalEntityId</td>
<td>The legal entity for the data import.</td>
</tr>
</tbody>
</table>

Output parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>string executionId</td>
<td>The execution ID of the data import. This is called as Job ID in the UI.</td>
</tr>
</tbody>
</table>

**NOTE**

ImportFromPackage() uses a batch to perform the import. Therefore, parallel processing rules must be used in Data management to perform parallel imports. ImportFromPackage() must not be called in parallel threads. Otherwise, it will fail.

Export APIs
The following APIs are used to export files (data packages).

**ExportToPackage**

The ExportToPackage API is used to initiate an export of a data package. This API is applicable to both cloud deployments and on-premises deployments.

- The export data project must be created before you call this API. If the project doesn’t exist, a call to the API returns an error.
- If change tracking has been turned on, only records that have been created or updated since the last run are exported. (In other words, only the delta is returned.)

```json
POST /data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.ExportToPackage
BODY
{
    "definitionGroupId":"<Data project name>",
    "packageName":"<Name to use for downloaded file.>",
    "executionId":"<Execution Id if it is a rerun>",
    "reExecute":<bool>,
    "legalEntityId":"<legal entity Id>"
}
```

Here is an example of a successful response.

```json
HTTP/1.1 200 OK
{
    "@odata.context":"https://<baseurl>/data/$metadata#Edm.String",
    "value":{
        "value":"<executionId>
    }
}
```

**Input parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>string definitionGroupId</td>
<td>The name of the data project for export.</td>
</tr>
<tr>
<td>string packageName</td>
<td>The name of the exported data package.</td>
</tr>
<tr>
<td>string executionId</td>
<td>The ID to use for the job. This is called as Job ID in the UI. If an empty ID is assigned, a new execution ID will be created.</td>
</tr>
<tr>
<td>bool reExecute</td>
<td>Set this parameter to True to run the target step. Otherwise, set it to False.</td>
</tr>
<tr>
<td>string legalEntityId</td>
<td>The legal entity for the data import.</td>
</tr>
</tbody>
</table>

**Output parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>string executionId</td>
<td>The execution ID of the data export. This is called as Job ID in the UI.</td>
</tr>
</tbody>
</table>

**GetExportedPackageUrl**

The GetExportedPackageUrl API is used to get the URL of the data package that was exported by a call to
**ExportToPackage.** This API is applicable to both cloud deployments and on-premises deployments.

```
POST /data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.GetExportedPackageUrl
BODY
{""executionId":"<Execution Id>"}
```

Here is an example of a successful response.

```
HTTP/1.1 200 OK
{
   "@odata.context":"https://<baseurl>/data/$metadata#Edm.String",
   "value":{
      "value":"https://<baseurl_id>.blob.core.windows.net/dmf/<uniqueFileName>?<SAS Token>"
   }
}
```

**Input parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>string executionId</td>
<td>The execution ID of the data project run. This is called as Job ID in the UI.</td>
</tr>
</tbody>
</table>

**Output parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>string BlobUrl</td>
<td>A blob URL that has an embedded SAS token. The URL can be used to download the exported data package.</td>
</tr>
</tbody>
</table>

**Status check API**

The following APIs are used to check status. They are used during both import flows and export flows.

**GetExecutionSummaryStatus**

The **GetExecutionSummaryStatus** API is used for both import jobs and export jobs. It's used to check the status of a data project execution job. This API is applicable to both cloud deployments and on-premises deployments.

```
POST /data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.GetExecutionSummaryStatus
BODY
{""executionId":"<executionId>"}
```

Here is an example of a successful response.

```
NOTE
The package needs to be created for the API to return the final execution status, such as Succeeded.
```

```
POST /data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.GetExecutionSummaryStatus
BODY
{""executionId":"<executionId>"}
```

Here is an example of a successful response.
HTTP/1.1 200 OK
{
   "@odata.context":"https://<baseurl>/data/$metadata#Edm.String",
   "value":"<executionStatus>"
}

### Input parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>string executionId</td>
<td>The execution ID of the data project run. This is called as Job ID in the UI.</td>
</tr>
</tbody>
</table>

### Output parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| DMFExecutionSummaryStatus executionStatus | The execution status. Here are the possible values:  
  - Unknown  
  - NotRun  
  - Executing  
  - Succeeded  
  - PartiallySucceeded  
  - Failed  
  - Canceled |

#### NOTE

The file in Blob storage will remain there for seven days. It will then be automatically deleted.

### Getting the list of errors

GetExecutionErrors can be used to get the list of errors in a job execution. The API takes the Execution ID as the parameter, and returns a set of error messages in a JSON list.

```plaintext
POST /data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.GetExecutionErrors
BODY
{"executionId":"<executionId>"}
```

### Import and export processes

The following illustration shows how the Data management package methods can be used to import data packages.
The following illustration shows how the Data management package methods can be used to export data packages.

A sample console application that showcases the data import and data export methods is available on GitHub. For more information, go to https://github.com/Microsoft/Dynamics-AX-Integration/tree/master/FileBasedIntegrationSamples/ConsoleAppSamples.
This topic describes the service endpoints that are available in Microsoft Dynamics 365 Finance. It also provides a comparison to the endpoints that are available in Microsoft Dynamics AX 2012.

List of services

The following table lists all the service endpoints, and compares the endpoints available for the application, and AX 2012.

<table>
<thead>
<tr>
<th>SERVICE ENDPOINT</th>
<th>AX 2012</th>
<th>FINANCE AND OPERATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document services (AXDs)</td>
<td>Yes</td>
<td>No – Replaced by data entities</td>
</tr>
<tr>
<td>SOAP-based metadata service</td>
<td>Yes</td>
<td>No – Replaced by REST metadata</td>
</tr>
<tr>
<td>SOAP-based query service</td>
<td>Yes</td>
<td>No – Replaced by OData</td>
</tr>
<tr>
<td>OData query service</td>
<td>Yes</td>
<td>No – Replaced by OData</td>
</tr>
<tr>
<td>SOAP-based custom service</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>JSON-based custom service</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>OData Service</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>REST Metadata Service</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This topic describes authentication for services, and the REST Metadata service. The following links provide detailed documentation for:

- Custom service development
- Open Data Protocol (OData)

Authentication

OData services, JSON-based custom services, and the REST metadata service support standard OAuth 2.0 authentication.

We currently support both Authorization Code Grant flow and Service to service calls using client credentials (shared secret or certificate).

Two kinds of application are supported in Microsoft Azure Active Directory (AAD):

- Native client application – This flow uses a user name and password for authentication and authorization.
- Web application (Confidential client) – A confidential client is an application that can keep a client password confidential to the world. The authorization server assigned this client password to the client application.

For more information, see:
Authorize access to web applications using OAuth 2.0 and Azure Active Directory

Troubleshoot service authentication issues

The following illustration describes how authorization must be configured for Authorization code grant flow.

And below is the illustration describes how authorization works for Service to service calls using client credentials (shared secret or certificate).

Register a web application with AAD

NOTE
These steps don’t have to be completed by all the people in your organization. Only one Azure Service Administrator user can add the application and share the client ID with the developers.

Prerequisite: You must have an Azure subscription and admin access to Azure Active Directory (Azure AD).

Before any clients can communicate with the services, they must be registered in (Azure AD). These steps will help you register an application with (Azure AD). The steps are explained in the Azure app registration training guide. For specific configuration in this process, the following additional information must be used in context.

Select Microsoft Dynamics ERP (Microsoft.ERP). If you search for Microsoft Dynamics ERP in the search field within Select an API it might appear to be unavailable. In that case, make sure that you search for the full name, as shown above. Under Delegated permissions, you must select, at a minimum, the following options:

- Access Dynamics AX Custom Service
- Access Dynamics AX data
Register your external application

1. In Finance and Operations apps, go to System administration > Setup > Azure Active Directory applications.

2. Select New.

3. Fill in the fields for the new record:
   - In the Client Id field, enter the application ID that you registered in Azure AD.
   - In the Name field, enter a name for the application.
   - In the User Id field, select an appropriate service account user ID. For this example, we have selected the Admin user. However, as a better practice, you should provision a dedicated service account that has the correct permissions for the operations that must be performed.

When you’ve finished, select Save.

You’ve now finished setting up the prerequisites. After the external application retrieves an Azure AD authentication token, it should now be able to use the token in an authorization HTTP header to make subsequent service calls via OData or SOAP, for example.

Client sample code

The following is C# sample code for getting a token from AAD. In this flow, the user will be presented with a consent form (for cross-tenant application) and a sign-in form.

```csharp
UriBuilder uri = new UriBuilder("https://login.windows.net/contoso2ax.onmicrosoft.com");
AuthenticationContext authenticationContext = new AuthenticationContext(uri.ToString());

//request token for the resource - which is the URI for your organization. NOTE: Important do not add a trailing slash at the end of the URI
AuthenticationResult authenticationResult = authenticationContext.AcquireToken("https://axdynamics1001aos.cloud.dynamics.com", clientId, redirectURI);

//this gets the authorization token, which needs to be passed in the Header of the HTTP Requests
string authenticationHeader = authenticationResult.CreateAuthorizationHeader();
```

To pass the user name and password without showing a pop-up, you can use the following overload of AcquireToken.

```csharp
UserCredential userCred = new UserCredential(username, password);
authenticationContext.AcquireToken("https://axdynamics1001aos.cloud.dynamics.com", clientId, userCred);
```

REST metadata service

The REST metadata service is a read-only service. In other words, users can make only GET requests. The main purpose of this endpoint is to provide metadata information for elements. It is an OData implementation.

This endpoint is hosted at *http://[/{baseURI}]/Metadata*.

Currently, this endpoint provides metadata for the following elements:
• **Labels** – Returns labels from the system. Labels have a dual pair key of language and ID, so that you can retrieve the value of the label.

  **Example:** [baseURI]/metadata/Labels(Id='@SVC\_ODataLabelFile:Label1',Language='en-us')

• **Data entities** – Returns a JSON-formatted list of all the data entities in the system.

  **Example:** [baseURI]/Metadata/DataEntities
This topic provides some tips for troubleshooting issues that involve service authentication.

When you troubleshoot service authentication issues, there are a few basic and common procedures that can help resolve the issues that are most often encountered. These procedures also provide a hands-on demonstration of how the authentication mechanism works. This topic includes instructions and also lists a few common issues that users have encountered so far.

**Inspect the JWT**

**Capture the JWT from an HTTP request**

2. Set up HTTPS capture to watch the HTTPS traffic from the client.
3. Find the Open Authorization (OAuth) JSON Web Token (JWT). It’s the value of the HTTP "Authorization" header without the "bearer" segment.

**Use a deserializer tool to look at the token contents**

1. Go to [https://jwt.io](https://jwt.io), and paste the JWT into the input panel.
2. View the contents in the form of name-value pairs. See the example that follows.
3. Verify that the following information is correct:
   - **"aud"** – The value corresponds to the Microsoft Azure Active Directory (Azure AD) resource concept. Here are some typical issues that involve "aud":
     - The "aud" segment of the JWT contains a URI that has a trailing slash.
     - The "aud" segment of the JWT contains a URI that uses an incorrect capitalization style. The URI must be all lowercase.
   - **"appid"** – The value corresponds to the Azure AD Native Client App ID (or Service App ID).
   - **"upn"** – The value corresponds to the user who is being authenticated through a Native Client App.

The following illustration shows an example of the contents of the JWT.
Review the event logs

You can also look at the event logs of the instance machine, if you have access to the virtual machine (VM).

1. Start Event Viewer by running the `eventvwr` command from the Run window.

2. Go to the following channels:
   - Application and Services Logs > Microsoft > Dynamics > AX-IntegrationServices > Channel:Operational (Microsoft-Dynamics-AX-IntegrationServices/Operational)
   - Application and Services Logs > Microsoft > Dynamics > AX-SystemRuntime > Channel:Operational (Microsoft-Dynamics-AX-SystemRuntime/Operational)

Other approaches

- For more information about how OAuth is configured, see Service endpoints overview.
- You can also try to call the service in parallel by using your own client code. The sample code that we published is available at https://github.com/Microsoft/Dynamics-AX-Integration.
- If the second method works, you can compare the JWTs from each method.

Known issues

AADSTS65001: The user or administrator hasn't consented to use the application

- The "aud" segment of the JWT might contain a URI that has a trailing slash. The slash must be removed.
- The "aud" segment of the JWT might contain a URI that uses an incorrect capitalization style. The URI must be all lowercase.
This topic provides information about Open Data Protocol (OData) and explains how you can use OData V4 to expose updatable views.

What is OData?

OData is a standard protocol for creating and consuming data. The purpose of OData is to provide a protocol that is based on Representational State Transfer (REST) for create, read, update, and delete (CRUD) operations. OData applies web technologies such as HTTP and JavaScript Object Notation (JSON) to provide access to information from various programs. OData provides the following benefits:

- It lets developers interact with data by using RESTful web services.
- It provides a simple and uniform way to share data in a discoverable manner.
- It enables broad integration across products.
- It enables integration by using the HTTP protocol stack.

For more information about OData, see the following webpages.

<table>
<thead>
<tr>
<th>TOPIC</th>
<th>WEBPAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OData standards</td>
<td>OData Version 4.01 documentation</td>
</tr>
<tr>
<td>OData: Data access for the web, the cloud, mobile devices, and more</td>
<td>OData in ASP.NET Web API</td>
</tr>
</tbody>
</table>

The public OData service endpoint enables access to data in a consistent manner across a broad range of clients. To see a list of all the entities that are exposed, open the OData service root URL. The URL for the service root on your system has the following format: [Your organization's root URL]/data

NOTE

OData actions added via extensions are currently not supported.

Addressing

The following table describes the resources and the corresponding URLs in the Fleet Management sample.

<table>
<thead>
<tr>
<th>RESOURCE</th>
<th>URL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service endpoint</td>
<td>[Your organization's root URL]/data/</td>
<td>The root service endpoint for OData entities</td>
</tr>
<tr>
<td>Entity collection</td>
<td>[Your organization's root URL]/data/Customers</td>
<td>The collection of all customers</td>
</tr>
<tr>
<td>Entity</td>
<td>[Your organization's root URL]/data/Customers(&quot;[key]&quot;)</td>
<td>A single entity from the entity collection</td>
</tr>
<tr>
<td>RESOURCE</td>
<td>URL</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Navigation property</td>
<td>[Your organization's root URL]/data/Customers(&quot;key&quot;)/Reservations</td>
<td>The navigation from a customer to that customer's reservations</td>
</tr>
<tr>
<td>Property</td>
<td>[Your organization's root URL]/data/Customers(&quot;key&quot;)/FirstName</td>
<td>The customer's first name</td>
</tr>
</tbody>
</table>

**OData services**

We provide an OData REST endpoint. This endpoint exposes all the data entities that are marked as `IsPublic` in the Application Object Tree (AOT). It supports complete CRUD (create, retrieve, update, and delete) functionality that users can use to insert and retrieve data from the system. Detailed labs for this feature are on the LCS methodology.

**NOTE**

When working with data entities using OData, all fields in the entity key must be provided to make a successful OData call.

Code examples for consuming OData services are available in the Microsoft Dynamics AX Integration GitHub repository.

**Supported features from the OData specification**

The following are the high-level features that are enabled for the OData service, per the OData specification.

- CRUD support is handled through HTTP verb support for POST, PATCH, PUT, and DELETE.

- Available query options are:
  - $filter
  - $count
  - $orderby
  - $skip
  - $top
  - $expand (only first-level expansion is supported)
  - $select

- The OData service supports serving driven paging with a maximum page size of 10,000.

For more information, see: OData actions that are bound to entities.

**Filter details**

There are built-in operators for $filter:

- Equals (eq)
- Not equals (ne)
- Greater than (gt)
- Greater than or equal (ge)
- Less than (lt)
- Less than or equal (le)
- And
• Or
• Not
• Addition (add)
• Subtraction (sub)
• Multiplication (mul)
• Division (div)
• Decimal division (divby)
• Modulo (mod)
• Precedence grouping ({})

You can also use the **Contains** option with $filter requests. It has been implemented as a wildcard character. For example: `http://host/service/EntitySet?$filter=StringField eq '\*retail\*'

The operators 'has' and 'in' are not supported.

For more information, see [OData operators](#).

**Batch requests**

Batch requests are supported in the OData service. For more information, see [OData batch requests](#).

**Metadata annotations**

/odata/$metadata provides annotations. EnumType is support in $metadata.

```xml
<EnumType Name="DimensionLedgerAccCategoryAccountType">
  <Member Name="ProfitAndLoss" Value="0"/>
  <Member Name="Revenue" Value="1"/>
  <Member Name="Expense" Value="2"/>
  <Member Name="BalanceSheet" Value="3"/>
  <Member Name="Asset" Value="4"/>
  <Member Name="Liability" Value="5"/>
  <Member Name="Equity" Value="6"/>
  <Member Name="Blank" Value="50"/>
</EnumType>
```

**Cross-company behavior**

By default, OData returns only data that belongs to the user’s default company. To see data from outside the user’s default company, specify the ?cross-company=true query option. This option will return data from all companies that the user has access to.

**Example:** `http://[baseURI]/data/FleetCustomers?cross-company=true`

To filter by a particular company that isn’t your default company, use the following syntax:

`http://[baseURI]/data/FleetCustomers?$filter=dataAreaId eq 'usr't&cross-company=true`

**Validate methods**

The following table summarizes the validate methods that the OData stack calls implicitly on the corresponding data entity.

<table>
<thead>
<tr>
<th>ODATA METHODS (LISTED IN THE ORDER IN WHICH THEY ARE CALLED)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Or</td>
</tr>
<tr>
<td>Not</td>
</tr>
<tr>
<td>Addition (add)</td>
</tr>
<tr>
<td>Subtraction (sub)</td>
</tr>
<tr>
<td>Multiplication (mul)</td>
</tr>
<tr>
<td>Division (div)</td>
</tr>
<tr>
<td>Decimal division (divby)</td>
</tr>
<tr>
<td>Modulo (mod)</td>
</tr>
<tr>
<td>Precedence grouping ({}))</td>
</tr>
<tr>
<td>ODATA</td>
</tr>
<tr>
<td>-------</td>
</tr>
</tbody>
</table>
| Create | 1. Clear()  
2. Initvalue()  
3. PropertyInfo.SetValue() for all specified fields in the request  
4. Validatefield()  
5. Defaultrow  
6. Validatewrite()  
7. Write() |
| Update | 1. Forupdate()  
2. Reread()  
3. Clear()  
4. Initvalue()  
5. PropertyInfo.SetValue() for all specified fields in the request  
6. Validatefield()  
7. Defaultrow()  
8. Validatewrite()  
9. Write() |
| Delete | 1. Forupdate()  
2. Reread()  
3. checkRestrictedDeleteActions()  
4. Validatedelete()  
5. Delete() |

**Exposing OData entities**

OData entities are based on the concept of an updatable view. When the `IsPublic` property for an updatable view is set to `TRUE`, that view is exposed as a top-level OData entity.

**Setting navigation properties between OData entities**

Links between OData entities are described by a navigation property. Navigation properties describe the navigation from one end of an association to the other end.

**Adding actions on OData entities**

Actions let you inject behaviors into the data model. To add actions, add a method to the updatable view, and decorate that method with specific attributes. Here is an example.

```csharp
[SysODataActionAttribute("CalcMaintenanceDuration", true)]
public int CalculateMaintenanceDuration()
{
    //do something
    return 0;
}
```

In this example, the `SysODataActionAttribute` class decorates the `CalculateMaintenanceDuration` method that is exposed as an action. The first argument of the attribute is the publicly exposed name of the action, and the second argument indicates whether this action is always available. Methods that are exposed as actions can return any primitive type or another public updatable view. After this method is exposed, it appears in the OData
Querying or browsing an OData endpoint

OData enables an SQL-like language that lets you create rich queries against the database, so that the results include only the data items that you want. To create a query, append criteria to the resource path. For example, you can query the Customers entity collection by appending the following query options in your browser.

<table>
<thead>
<tr>
<th>URL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Your organization's root URL]/data/Customers</td>
<td>List all the customers.</td>
</tr>
<tr>
<td>![Your organization's root URL]/data/Customers?$top=3</td>
<td>List the first three records.</td>
</tr>
<tr>
<td>![Your organization's root URL]/data/Customers?$select=FirstName,LastName</td>
<td>List all the customers, but show only the first name and last name properties.</td>
</tr>
<tr>
<td>![Your organization's root URL]/data/Customers?$format=json</td>
<td>List all the customers in a JSON format that can be used to interact with JavaScript clients.</td>
</tr>
</tbody>
</table>

The OData protocol supports many similar filtering and querying options on entities. For the full set of query options, see Windows Communication Foundation.

Using Enums

 Enums are under namespace Microsoft.Dynamics.DataEntities. Enums can be included in an OData query is by using the following syntax.
Authentication

Tips and tricks

Run multiple requests in a single transaction

The OData batch framework uses changesets. Each changeset contains a list of requests that should be treated as single atomic unit. In other words, either all the requests are run successfully or, if any request fails, none of the requests are run successfully. The following example shows how to send a batch request that has a list of requests in a single changeset.

The SaveChangesOptions.BatchWithSingleChangeset option in SaveChanges() helps guarantee that all requests are bundled into a single changeset.

```csharp
public static void CreateProductColors(Resources context)
{
    var productColorsCollection = new DataServiceCollection<ProductColor>(context);
    var color1 = new ProductColor();
    productColorsCollection.Add(color1);
    color1.ColorId = "New Color1"; // set any other properties needed
    var color2 = new ProductColor();
    productColorsCollection.Add(color2);
    color2.ColorId = "New Color2"; // set any other properties needed
    context.SaveChanges(SaveChangesOptions.BatchWithSingleChangeset);
}
```

Prevent unset records from being posted when you use an OData client

When you create a new record by using an OData client, as shown in example 1, properties that aren't set are included in the body of the request, and default values are assigned to them. To prevent this behavior and post only properties that are set explicitly, use the SaveChangesOptions.PostOnlySetProperties option in SaveChanges(), as shown in example 2.

Example 1
public static void CreateVendor/Resources(context) {
    var vendorCollection = new DataServiceCollection/Vendor>(context);
    var vendor = new Vendor();
    vendorCollection.Add(vendor);
    // set properties
    context.SaveChanges();
}

Example 2

public static void CreateVendor/Resources(context) {
    var vendorCollection = new DataServiceCollection/Vendor>(context);
    var vendor = new Vendor();
    vendorCollection.Add(vendor);
    // set properties
    // Save specifying PostOnlySetProperties flag
    context.SaveChanges(SaveChangesOptions.PostOnlySetProperties);
}

Handling duplicate names between enums and entities in metadata
There are instances where enums and entities share the same name. This name duplication results in OData client code generation errors. To recover from this error, the helper code in GitHub can be used to identify duplicate name instances that must be removed. The generated metadata document can be used for further processing of the OData logic on the client side.

Array fields
OData does not support array fields in entities. This must be taken into consideration when designing entities that will be used with OData.

After restarting AOS, the first OData call may take a long time to process
The first OData call processed by an AOS that was restarted may take a long time to process because the metadata is not being cached. This latency can be avoided by warming up OData on AOS startup. For more details, see Build OData metadata cache when the AOS starts.
You can develop custom services for Finance and Operations. When a developer writes a custom service under a service group, the service group is always deployed on two endpoints:

- SOAP endpoint
- JSON endpoint

**SOAP-based custom service**

SOAP-based services remain the same as they were in Dynamics AX 2012.

Code examples for consuming custom services using SOAP are available in the Microsoft Dynamics AX Integration GitHub repository.

**Key changes**

- All the service groups under the **AOTService group** node are automatically deployed.
- All services that must be deployed must be part of a service group.

**Example endpoint for a dev environment**

https://usnconeboxax1aos.cloud.onebox.dynamics.com/soap/services/UserSessionService?wsdl

**Example endpoint for a non-dev environment**

https://<baseurl>/soap/services/UserSessionService?wsdl

For more information about custom services, see:

- **Using Custom Services [AX 2012] (TechNet)**
- **Walkthrough: Exposing an X++ Class as a Data Contract (TechNet)**

**JSON-based custom service**

This feature enables X++ classes to be consumed as JSON services. In other words, the return data set is in JSON format. JSON, which stands for JavaScript Object Notation, is a compact, lightweight format that is commonly used to communicate data between the client and the server.

The JSON Endpoint is at

https://host_uri/api/services/service_group_name/service_group_service_name/operation_name

**Example**


Code examples for consuming JSON services are available in the Microsoft Dynamics AX Integration GitHub repository.
Recurring integrations do the following things:

- It builds on data entities and the Data management framework.
- It enables the exchange of documents or files between Finance and Operations and any third-party application or service.
- It supports several document formats, source mapping, Extensible Stylesheet Language Transformations (XSLT), and filters.

- It uses secure REST application programming interfaces (APIs) and authorization mechanisms to receive data from, and send data back to, integration systems.

Authorization for the integration REST API

The integration REST API uses the same OAuth 2.0 authentication model as the other service endpoints. Before the integrating client application can consume this endpoint, you must create an application ID in Microsoft.
Azure Active Directory (Azure AD) and give it appropriate permission to the application. When you create and enable a recurring job, you're prompted to enter the Azure AD application ID that will interact with that recurring job. Therefore, be sure to make a note of the application ID.

**NOTE**
This feature is not supported with Dynamics 365 Finance + Operations (on-premises).

### Set up a data project and recurring data jobs

#### Create a data project

1. On the main dashboard, select the **Data management** tile to open the **Data management** workspace.
2. Select the **Import** or **Export** tile to create a new data project.

   **NOTE**
   If you have an existing data project, select **Load project** on the card for any data project on the **Data projects** tab.

3. Enter a valid job name, data source, and entity name.
4. Upload a data file for one or more entities. Make sure that each entity is added, and that no errors occur.

   **NOTE**
   You can select each entity data card to set up, review, or modify field maps, and to set up XSLT-based transforms that must be applied to inbound data. For export data projects, the entity card also shows a filter link, so that you can set up filters to filter data. Currently, all recurring data jobs in a data project use the same filter.

5. Select **Save**.

#### Create a recurring data job

1. On the **Data project** page, select **Create recurring data job**.
2. Enter a valid name and a description for the recurring data job.
3. On the **Set up authorization policy** tab, enter the application ID that was generated for your application, and mark it as enabled.
4. Expand **Advanced options** tab, and specify either **File** or **Data package**.
   - **File** – Your external integration will push individual files so that they can be processed via this recurring data job. In this case, the format of the file that is expected is the same as the format that was specified when the entity was added to the data project.
   - **Data package** – You can push only data package files for processing. A data package is a new format that lets you submit multiple data files as a single unit that can be used in integration jobs.
   - **Process messages in order** – You can enable this option to force sequential processing of incoming files in an import scenario. This option is only applicable to files and not data packages.
5. Select **Set processing recurrence**, and then, in the **Define recurrence** dialog box, set up a valid recurrence for your data job.
6. Optional: Select **Set monitoring recurrence**, and set up a monitoring recurrence.
NOTE
Currently, monitoring recurrence enables load monitoring only on the queue for your recurring data job. No additional policies are supported via this service. You can use this feature to fine-tune the processing recurrence as the load demand requires.

7. Select **OK**, and then select **Yes** in the confirmation message box.

**Manage recurring data jobs**

1. In the **System administration** workspace (not the **System administration** module), select the **Data Management IT** workspace.

2. In the workspace, on the **Recurring data job** tab, select the recurring job to view the details. The **Manage scheduled data jobs** page contains a grid that lists any messages that are waiting in the queue. Therefore, you can monitor messages and the processing status.

**Submitting data to recurring data jobs**

You can use integration REST endpoints to integrate with the client, submit documents (import), or pull available documents for download (export). These endpoints support OAuth.

**Integration REST APIs**

The following set of APIs is used to exchange data between the integration client and the application.

**API for import (enqueue)**

Make an HTTP POST call against the following URL.

```
https://<base URL>/api/connector/enqueue/<activity ID>?entity=<entity name>
```

In the message body, you can pass the data as a memory stream.

**Example**

```
POST https://usncax1aos.cloud.onebox.dynamics.com/api/connector/enqueue/%7B6D31E09F-0249-459F-94F0-AAD9C2C47B64%7D?entity=Customer%20Groups
```

To get the activity ID, on the **Manage scheduled data jobs** page, in the **ID** field, copy the globally unique identifier (GUID).
**API for export (dequeue)**

To return a data package that contains all the data entities that were defined in the data project, and that the client application can unzip and consume, use the following structure.

```
https://<base URL>/api/connector/dequeue/<activity ID>
```

**Example**

```
GET https://usncaxlaos.cloud.onebox.dynamics.com/en/api/connector/dequeue/%7BC03BB937-09ED-46DE-86EE-452807D7E373%7D
```

After the client downloads the data, an acknowledgment must be sent back to the application, so that you can mark the data as received.

In cases when there was no file uploaded to the blob, the dequeue API will return a response indicating as such.

**API for acknowledgment**

Use the following API.

```
NOTE
The body of the response of /dequeue must be sent in the body of the /ack POST request.
```

```
https://<base URL>/api/connector/ack/<activity ID>
```

**Example**

```
POST https://usncaxlaos.cloud.onebox.dynamics.com/en/api/connector/ack/%7BC03BB937-09ED-46DE-86EE-452807D7E373%7D
```

**NOTE**

Until a message is successfully acknowledged, the same message will become available to dequeue every 30 minutes. In cases when a message is being dequeued more than one time, the dequeue response will send the last dequeued date time. This will be blank for the first dequeue of a message. It is important to ensure that a message is successfully acknowledged to prevent a repeated download of the same message. When an acknowledgement fails, having re-try logic to acknowledge the failure is recommended.

**API for getting message status**

The API to get the status of a message is available as of hotfix KB 4058074 for Platform update 12. This API is particularly useful in import scenarios to determine if a message has been successfully processed. A message is created when the enqueue process is completed. If the message returns a failed status, you can set your integration app to retry or take another action.

**Example**

```
POST /data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.GetMessageStatus
BODY
{
    "messageId":"<string>
}
```
The following table lists the possible status values.

<table>
<thead>
<tr>
<th>VALUE</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enqueued</td>
<td>The file has been successfully enqueued to blob storage</td>
</tr>
<tr>
<td>Dequeued</td>
<td>The file has been successfully dequeued from blob storage</td>
</tr>
<tr>
<td>Acked</td>
<td>The exported file has been acknowledged to be downloaded by the external application</td>
</tr>
<tr>
<td>Preprocessing</td>
<td>The import/export operation is pre-processing the request</td>
</tr>
<tr>
<td>Processing</td>
<td>The import/export operation is in process</td>
</tr>
<tr>
<td>Processed</td>
<td>The import/export operation completed successfully</td>
</tr>
<tr>
<td>PreProcessingError</td>
<td>The import/export operation failed in the pre-processing stage</td>
</tr>
<tr>
<td>ProcessedWithErrors</td>
<td>The import/export operation completed with errors</td>
</tr>
<tr>
<td>PostProcessingFailed</td>
<td>The import/export operation failed during post-processing</td>
</tr>
</tbody>
</table>

**NOTE**

The file in the blob storage will remain in the storage for seven days, after which it will be automatically deleted.

**API to get the list of execution errors**

GetExecutionErrors can be used to get the list of errors in a job execution. The API takes the Execution ID as the parameter, and returns a set of error messages in a JSON list.

```
POST /data/DataManagementDefinitionGroups/Microsoft.Dynamics.DataEntities.GetExecutionErrors
BODY
{"executionId":"<executionId>"}
```

**Tips and tricks**

**Viewing the batch job status for recurring integrations from the Data management workspace**

Recurring integration data jobs run in batch mode. If a recurring job fails, you must investigate the instance of the batch job as part of the troubleshooting process. To make this investigation easier, click Manage messages to get to the Process status for recurring data job page, which now shows the status of the batch job.

The batch job status is retrieved asynchronously from the batch framework for the specified recurring data job. To see the most up-to-date batch job status, select Get batch status, and then refresh the page.

**NOTE**

If the record for the batch history is deleted, the status for the batch job on the Processing status for recurring data job page will be blank.
Preventing uploads when there are no records

When you use recurring exports you can choose not to upload a generated file or package if the total record count in that file or package is 0 (zero).

You can set the Prevent upload when zero records option either when you configure a recurring export job or after a job has been created. This option is available only when you use files or packages as data sources.

Create recurring data job

Your implementation might include runs of recurring jobs where files or packages were uploaded. Your implementation might also include runs where no files or packages were uploaded, because there was nothing to upload. If you suspect that a file that should have been uploaded wasn’t uploaded, or that a file that should not have been uploaded was uploaded, you can use the Manage messages page for the recurring export job to help with the debugging process.

NOTE

These features were added in Microsoft Dynamics 365 for Finance and Operations, Enterprise edition platform update 12. Jobs that were run before you upgraded to Platform update 12 won’t have values in the following columns.

The Total records exported column shows the total count of records that were exported. A value of 0 (zero) indicates that no records were exported to the file or included in the package.

The File uploaded successfully column contains a check mark if the file or the package was uploaded.
Http vs Https

The dequeue API returns HTTP instead of HTTPS. This behavior can be seen in application environments that use a load balancer, such as production environments. (You cannot see the behavior in one box environments). We recommend that you change the URI scheme to HTTPS in the middleware application that is trying to dequeue from the application.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message Status</th>
<th>Source Status</th>
<th>Total Records Requested</th>
<th>Total Records Successfully Retrieved</th>
<th>Processing Started Date Time</th>
<th>Processing Completed Date Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>090E3D56-C8C4-46CA-8601-440144A46074</td>
<td>Processed</td>
<td>Created</td>
<td>127</td>
<td>yes</td>
<td>12/5/2017 02:39:35 PM</td>
<td>12/5/2017 02:39:40 PM</td>
</tr>
<tr>
<td>090E3D56-C8C4-46CA-8601-440144A46074</td>
<td>Processed</td>
<td>Created</td>
<td>127</td>
<td>yes</td>
<td>12/5/2017 02:39:35 PM</td>
<td>12/5/2017 02:39:40 PM</td>
</tr>
<tr>
<td>090E3D56-C8C4-46CA-8601-440144A46074</td>
<td>Processed</td>
<td>Created</td>
<td>127</td>
<td>yes</td>
<td>12/5/2017 02:39:35 PM</td>
<td>12/5/2017 02:39:40 PM</td>
</tr>
<tr>
<td>090E3D56-C8C4-46CA-8601-440144A46074</td>
<td>Processed</td>
<td>Created</td>
<td>127</td>
<td>yes</td>
<td>12/5/2017 02:39:35 PM</td>
<td>12/5/2017 02:39:40 PM</td>
</tr>
<tr>
<td>090E3D56-C8C4-46CA-8601-440144A46074</td>
<td>Processed</td>
<td>Created</td>
<td>127</td>
<td>yes</td>
<td>12/5/2017 02:39:35 PM</td>
<td>12/5/2017 02:39:40 PM</td>
</tr>
<tr>
<td>090E3D56-C8C4-46CA-8601-440144A46074</td>
<td>Processed</td>
<td>Created</td>
<td>127</td>
<td>yes</td>
<td>12/5/2017 02:39:35 PM</td>
<td>12/5/2017 02:39:40 PM</td>
</tr>
<tr>
<td>090E3D56-C8C4-46CA-8601-440144A46074</td>
<td>Processed</td>
<td>Created</td>
<td>127</td>
<td>yes</td>
<td>12/5/2017 02:39:35 PM</td>
<td>12/5/2017 02:39:40 PM</td>
</tr>
<tr>
<td>090E3D56-C8C4-46CA-8601-440144A46074</td>
<td>Processed</td>
<td>Created</td>
<td>127</td>
<td>yes</td>
<td>12/5/2017 02:39:35 PM</td>
<td>12/5/2017 02:39:40 PM</td>
</tr>
<tr>
<td>090E3D56-C8C4-46CA-8601-440144A46074</td>
<td>Processed</td>
<td>Created</td>
<td>127</td>
<td>yes</td>
<td>12/5/2017 02:39:35 PM</td>
<td>12/5/2017 02:39:40 PM</td>
</tr>
<tr>
<td>090E3D56-C8C4-46CA-8601-440144A46074</td>
<td>Processed</td>
<td>Created</td>
<td>127</td>
<td>yes</td>
<td>12/5/2017 02:39:35 PM</td>
<td>12/5/2017 02:39:40 PM</td>
</tr>
</tbody>
</table>

Successfully. If the file wasn’t uploaded because of an error, or because there were no records, the column will be blank.
At https://github.com/Microsoft/Dynamics-AX-Integration, Microsoft provides sample code for consuming services. However, there are many scenarios where the other endpoint in an integration might not use a Microsoft stack. Even when the other endpoint does use, for example, the Open Data Protocol (OData) client code that Microsoft makes available, you might find it useful to perform the following actions:

- Explore and analyze how an interaction's messages are constructed.
- Test the response of a service to a well-known request.
- Determine how exceptions will appear to the other endpoint.

Many frequently used tools that will help you perform these actions are available. This topic isn't an endorsement of any tool. Although it provides examples that use some frequently used software utilities, the principles should broadly apply to other, similar tools.

**Prerequisites**

Before you can test a service by using an external application, you must register the application in Microsoft Azure, and in Finance and Operations.

For details, see:

- Register an application with AAD
- Register your external application

**Query OData by using Postman**

Postman (https://www.getpostman.com/postman) is a tool that is often used to interact with RESTful services (such as OData) in scenarios that involve the development and testing of application programming interfaces (APIs). This procedure isn't an endorsement of Postman, and other similar tools are available. However, we are using Postman to illustrate the concepts and messages that are involved when you use OAuth to authenticate with Azure AD, and then make OData requests to and receive responses from the application.

1. Start Postman.
2. In the upper-right corner, select the gear button, and then select **Manage environments** to create or update an environment.
3. Enter a name for the environment, and then select **Bulk Edit**.
4. Enter key-value pairs as shown in the following table. Enter one pair per line, and separate the key and value by using a colon (:).

<table>
<thead>
<tr>
<th>KEY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>tenant_id</td>
<td>The Azure tenant ID that you looked up during the setup of prerequisites</td>
</tr>
<tr>
<td>client_id</td>
<td>The Azure AD application ID that you registered during the setup of prerequisites</td>
</tr>
</tbody>
</table>
5. To verify that the key-value pairs can be parsed correctly, select **Key-Value Edit**, and review the results.

6. Close the environment page.

7. In the field to the left of the gear and eye buttons, select the new or updated environment.

8. To retrieve an Azure AD token, create a POST request that has a URL in the format
   \( https://login.microsoftonline.com/[\text{tenant ID}]\text{/oauth2/token} \).

   You can use a URL parameter that refers to the `tenant_id` environment variable, such as
   \( https://login.microsoftonline.com/:tenant_id\text{/oauth2/token} \).

9. On the **Body** tab, add body elements as request parameters that refer to the environment variables that you created earlier. Select **Bulk Edit**, enter the keys from the previous table, enter a colon (:), and then enter the key name again but enclose it in double braces ({{}}). Enter one request parameter per line. For example, enter `grant_type:{{grant_type}}`. Here is an example.

10. On the **Tests** tab, create a test that validates that the response is reasonable, and that stores the returned authorization token in an environment variable. Here is an example.
11. Select **Save**, enter a name and collection for the request, and then select **Save** again.

12. Select **Send** to make the authorization request. The **Body** tab should now contain an Azure AD token together with other response details.

13. Because of the test code, the token is now in an environment variable. You can see that the token is an environment variable by selecting the **Environment quick look** button (the eye button).

14. Create a request to perform create, read, update, or delete (CRUD) operations on the desired data entity.
via the OData service. Create the URL according to your requirements. For more information, see **Open Data Protocol (OData)**. You might find it useful to parameterize the request by using a variable that is stored in the environment, as shown earlier. The following example of a GET query uses a **Customer Account** parameter. The query returns name and address details for the customer account that is specified in the environment variable. Note that special characters must be correctly URL-encoded.

```
https://[Finance and Operations instance URL]/data/Customers?
$format=json&$filter=CustomerAccount%20eq%20{{custAccount}}&$select=CustomerAccount,Name,AddressDescription,FullPrimaryAddress
```

15. Add an Authorization header that refers to the authorization token that was retrieved earlier and stored in the **bearerToken** environment variable. The token must be prefixed by **Bearer** in the header.

16. Create a test to help validate the response. The following example tests that non-empty, JSON-formatted data is returned in the response body.

```
var json = JSON.parse(responseBody);
tests["Get customer info"] = !json.error && responseBody != '' && responseBody !== '{}';
```

17. Save and send the request, and then verify the result. You must ensure that the user account being used is set to a default company that has data. Alternatively, you can also specify `cross-company=true` as the query parameter in the OData request.
In our example, we have now successfully authenticated and then used the OData service to read a customer record.

Query the SOAP custom service in your application by using SoapUI

SoapUI (https://www.soapui.org/) is a tool that is often used to interact with SOAP and REST web services in scenarios that involve API development and testing. This procedure isn't an endorsement of SoapUI, and other similar tools are available. However, we are using SoapUI to illustrate the concepts and messages that are involved when you use OAuth to authenticate with Azure AD, and then make SOAP requests to and receive responses.

1. Start SoapUI, and select the SOAP button to create a project.

2. Complete the information for the project:
   - In the Project Name field, enter a name for the project.
   - In the Initial WSDL field, enter the service address, and add the suffix ?wsdl. (The service address should be in the format [Finance and Operations instance base URL]/soap/services/[service group name].) For more information, see the Services home page.
     
     For example, we are querying the user session service at the URL https://[Finance and Operations base URL]/soap/services/UserSessionService?wsdl.

   - Select the Create sample requests for all operations? check box.

     Because you selected to create sample requests, one sample request is created for each service operation that is available.

3. Right-click the new project, and then select New TestSuite to create a test suite. This test suite will generate a POST request for an Azure AD authorization token.

4. Right-click the test suite, and then select New TestCase.

5. Expand the test case, right-click Test Steps, select Add Step, and then select HTTP Request.

6. Enter a name for the request, and then select OK.

7. Enter a name for the test step. The endpoint that you should use for the POST request is
8. Use the plus sign (+) button next to Parameters to add the following values.

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>grant_type</td>
<td>client_credentials</td>
</tr>
<tr>
<td>client_id</td>
<td>The application ID from the Azure AD application registration</td>
</tr>
<tr>
<td>client_secret</td>
<td>The secret key value from the Azure AD application registration</td>
</tr>
<tr>
<td>resource</td>
<td>The URL of the instance without the trailing ‘/’</td>
</tr>
</tbody>
</table>

9. To make sure that the parameters are in the POST body, select Post QueryString, and then select Play. An access token should be returned in the response pane. The values will be most readable if you use the JSON response tab. Copy the access token so that you can use it in the authorization header of subsequent requests.

10. Go back to the first request node under the GetUserSessionInfo SOAP sample request. In the request pane on the left, select the plus sign (+) button to add a header that is named Authorization. Paste the access token into the Value field, and add the prefix Bearer.

11. The sample requests that SoapUI creates won’t work unless you modify them. You must edit the call context and body so that they are consistent with the schema for what you’re trying to do.

For our simple scenario, you can edit the optional call context elements so that they are null-valued. Insert a forward slash (/) before the greater than sign (>) in the opening tags. Then comment out the question marks (?) and the closing tags by using the standard <!--...--> syntax to delimit the start and end of the comments. (Question marks aren’t valid content for the XML schema.) Alternatively, you can just delete the question marks (?) so that the context elements are empty.

12. The SOAP request is now ready. Select Play, and validate the result on the right.

In our example, we have now successfully authenticated and then queried UserSessionService via SOAP.
The application connector allows Microsoft Power Automate, Power Apps, Data Integrator, and Logic Apps to integrate with Finance and Operations. An external application can use the available trigger and actions to integrate with them.

**IMPORTANT**

The Application Connector cannot be used for integrations with Dynamics 365 Finance + Operations (on-premises) instances.

**Prerequisites**

We recommend that you read the following topics as a prerequisite to familiarize yourself with connectors before proceeding further:

- Connectors
- Data management package REST API
- Open Data Protocol (OData)
- Recurring integrations

**Triggers**

Business events are exposed using the trigger *When a business event occurs*. For detailed information about business events, refer to Business events in Microsoft Power Automate and Business events.

**Actions**

This section describes the actions that are available in the connector.

**Get a record**

This action can be used to fetch a record for a specific data entity from the target instance.

*Instance* refers to the URL of the target instance of the application to which the connector must connect. The expected value is to enter the URL without the "https://" prefix or choose one from the drop-down menu. This lists of all the environments that are deployed in the Azure Active Directory tenant for the user account that was used to sign in to the specific client like Power Automate, Power Apps, or Logic App.

*Entity name* refers to the data entity from which the record must be fetched. The drop-down menu shows the list of data entities from the target environment.

*Object ID* refers to the primary keys fields that must be specified to uniquely identify the record that must be fetched. The values must be specified as a comma-separated list of values in the order that is defined in the entity.

**Create a record**

This action can be used to create data records for a data entity.

*Instance* refers to the URL of the target instance to which the connector must connect. The syntax for this value
is to enter the URL without the ‘https://’ prefix or choose one from the drop-down menu. This lists of all the environments that are deployed in the Azure Active Directory tenant for the user account that was used to sign in to the specific client like Power Automate, Power Apps, or Logic App.

*Entity name* refers to the data entity in which the record must be created. The dropdown menu shows the list of data entities from the target environment.

Based on the selected data entity, the list of fields displayed will vary.

**Update a record**

This action can be used to update an existing data record for a data entity. The usage is the same as the create a record action.

**Delete a record**

This action can be used to delete an existing data record for a data entity. The usage is the same as the get a record action.

**Execute action**

This action can be used to invoke methods on a data entity to perform a business action.

*Instance* refers to the URL of the target instance to which the connector must connect. The syntax for this value is to enter the URL without the ‘https://’ prefix or choose one from the drop-down menu. This lists of all the environments that are deployed in the Azure Active Directory tenant for the user account that was used to sign in to the specific client like Power Automate, Power Apps, or Logic App.

*Action* refers to the method on the data entity that must be executed. Based on the selected method, the list of fields displayed will vary. These fields represent the parameters for the selected method.

**Get list of entities**

This action can be used to get the list of entities for further use in the app that is being developed.

*Instance* refers to the URL of the target instance to which the connector must connect. The syntax for this value is to enter the URL without the ‘https://’ prefix or choose one from the drop-down menu. This lists of all the environments that are deployed in the Azure Active Directory tenant for the user account that was used to sign in to the specific client like Power Automate, Power Apps, or Logic App.

**List items present in the table**

This action can be used to get the list of records from an entity. This action supports cross-company reading of data.
This topic provides a brief overview of the mechanics of synchronous and asynchronous integration.

Synchronous services

Synchronous integrations are relatively straightforward. Any entity that has **is public** enabled is automatically available as a service application programming interface (API) in the following URL:

https://[BaseURL]/Data/<<Data Entity Public Collection Name>>

Currently, OData protocol is used to expose endpoints where all public-enabled entities can be interacted with.

**Supported protocol:** OData V4.0

**Data format:** JSON

**Metadata URL:** https://[BaseURL]/Data/$metadata

Data import/export and recurring integration scenarios

Integration through the data management platform provides more capabilities and higher throughput for inserting/extracting data through entities. Typically, data goes through three phases in this integration scenario:

- **Source** – These are inbound data files or messages in the queue. Typical data formats include CSV, XML, and tab-delimited.
- **Staging** – These are automatically generated tables that map closely to the data entity. When **Data management enabled** is true, staging tables are generated to provide intermediary storage. This enables the framework to do high-volume file parsing, transformation, and some validations.
- **Target** – This is the data entity where data will be imported.

The following diagram shows an inbound flow.

Known limitations in data import/export
When you import text files, string sizes are limited to 32,768 characters. If there is a string larger than this, the imported string will be truncated. This is a limitation in the underlying implementation and is due to SQL Server Integration Services (SSIS).

If you need to import strings that are larger than 32,768 characters, we suggest that you use container entity fields.

For more information, watch the FastTrack Tech Talk video: Dynamics 365 for Operations – Tech Talk: Integration.
This tutorial shows how to develop data entities in Microsoft Visual Studio and then use them for data migration.

This tutorial is broken out into two sections and four exercises. In the first section, you will build a **Project Category** entity in Visual Studio. You will then use this entity to export data. In the second section, you will use **Customer Groups** and **Customers** entities to import multiple sets of files by using the new Data Import/Export Framework.

**NOTE**

This tutorial is designed to be slightly more challenging than [Build and consume data entities](#). Instead of providing a step-by-step guide, it has scenario exercises and describes the expected outcomes. The assumption is that you've already familiarized yourself with entities.

**Prerequisites**

This tutorial requires that you access the environment by using Remote Desktop. You must be provisioned as an administrator on the instance.

**Base URL**

Throughout this tutorial, "base URL" refers to the base URL of the instance.

- In the cloud environment, you obtain the base URL from Microsoft Dynamics Lifecycle Services (LCS).
- On a local virtual machine (VM), the base URL is [https://usnconeboxax1aos.cloud.onebox.dynamics.com](https://usnconeboxax1aos.cloud.onebox.dynamics.com).

## Developing an entity in Visual Studio and enabling it for data export

**Business problem**

You're developing a new solution for a Project module. As part of your implementation, you must represent the data from project categories, so that this data can be imported into the system or exported from it. To accomplish this goal, you will first build an entity for the project category and then use the export functionality to test data extraction.

**Exercise 1: Build a Project Category entity**

In this exercise, you will build an entity, **Project Category**, that uses the ProjCategory table as its primary data source. This entity has the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity AOT name</td>
<td>ProjectCategoryEntity</td>
</tr>
<tr>
<td>Label</td>
<td>Project Categories</td>
</tr>
<tr>
<td>Entity category</td>
<td>Reference</td>
</tr>
</tbody>
</table>
The entity also has the following fields:

- ActiveInJournals
- CategoryGroup
- Category
- TransactionType
- CategoryName
- Worker
- CustomerPaymentRetention
- IndirectCostComponent
- ItemSalesTaxGroup
- ServiceCode
- Absence

**Steps**

1. In Visual Studio, create a new application project.

2. In Solution Explorer, select the project, and then right-click **Properties**.

3. Specify the following project properties, and then click **OK**.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Application Suite</td>
</tr>
<tr>
<td>Company</td>
<td>USSI</td>
</tr>
<tr>
<td>Synchronize Database on build</td>
<td>True</td>
</tr>
</tbody>
</table>

4. From the project, right-click **Add > New item**.

5. Select **Data Model > Data Entity** as your new item.

6. Enter a name, and then click **Add** to start the **Data Entity** wizard.

7. On the first page of the wizard, specify the set of properties for the entity by using the table earlier in this exercise. Then click **Next**.
Exercise 2: Export a limited set of data by using a sample file mapping and filters

In this exercise, you will use the Project Category entity that you just built to export data. To export only a subset of the data, you will use a sample file mapping and filters. The exported data will be in XML format.

Steps

1. After you’ve finished building the Project Category entity, start the client.

2. Change the company to USSI.

3. In the Data management workspace, click Export to begin data extraction.

4. Enter the export details, such as entity name and target data format.

Expected outcome

- In Visual Studio, the following artifacts will appear in the project after you’ve completed the Data Entity wizard.

- Right-click the data entity, and then select Open table browser.

**NOTE**

Make sure that your company is set to USSI.
Use the following file as the sample file format for XML: `ProjectCategoryExport_Sample`.

Open this file in a text editor, and save it as an XML file. If the sample file mapping isn't valid, there is an incorrect field name in the entity. Fix either the entity or the sample file to continue.

5. Click **Filter**, and then specify **Project** as the filter criterion, so that only limited data is exported.

6. In the **Export** dialog box, click **OK**.

**Expected outcome**
- Fifteen records are successfully exported.
- The output is similar to the following file: `ProjectCategoryExport_Output` (Open the file in a text editor to verify this outcome.)

**Migrating data in multiple files by using the Data Import/Export Framework**

**Business problem**

You're implementing a new environment. As part of this implementation, you want to migrate some legacy customer data. The data is contained in two sets of files, each of which has data for the **Customers** and **Customer Groups** entities. This migration is slightly complex, because some columns in the data files don't
Exercise 3: Create a data project and import multiple files

In this exercise, you will import two files into the USRT company by using the new Data Import/Export Framework. These files need to be imported in sequence by using a single data project. The Customers entity has a reference to the Customer Groups entity. Because the Customers1 file doesn't map correctly to the Customers entity, you will receive an error when you upload the file. Therefore, to complete the import process, you will have to provide the correct column mappings for the Customers entity.

Steps
1. Open the following files in Microsoft Excel, and save them as CSV files in your local directory:
   - Customers1
   - CustomerGroups1
2. In the client, change the company to USRT.
3. From the User dashboard, open the Data Management workspace.
4. Click Import to configure a new data project.
5. Enter the project details, such as the name and file format.
6. For each file, select an entity, and then upload the data file.
7. Because the Customer1.csv file doesn't map correctly to the Customers entity, you will receive an error when you upload the file. After the file is uploaded, click View mappings to fix the column mappings for the Customers entity.

   TIP
   CustomerAccount is required in the entity during import. It is mapped from AccountNum in the source file. Address fields are optional for the import.

8. When you’ve finished, click the Back button in your browser to return to the data project.

Expected outcome
Four updates and 23 inserts are successfully imported and the Execution summary page shows the results.

Exercise 4: Re-import by using an existing data project and manage data in staging

In this exercise, you will use a new set of files to import data through the existing data project. Customers2 contains new and updated data for the Customers entity. CustomerGroups2 contains updated data for the Customer Groups entity. Customers2 contains some error records. You will fix these errors in staging, validate the data, and then push it to the target to complete the import.

Steps
1. In the Data management workspace, select the existing data project, and the click Re-import. By using the re-import functionality, you can preserve your previous settings for the data project and use new files for the import. However, if you click Reload data project and upload new files instead, the previous mappings will be overridden.
2. Open the following files in Excel, and save them as CSV files in your local directory:
   - Customers2
   - CustomerGroups2
3. Upload the new files for each entity, and then click **Import now**.

4. On the **Status** page, click **View execution log** to investigate the errors.

5. On the **Status** page, click **View staging** to view the data in a staging table. This view will also show records that have errors.

6. Click **Edit** to fix records that have errors. (The **Customer Group** value for records DM10221 and DM1022 isn't valid.)

7. Select the records that you fixed, and then click **Validate**. Refresh the page to verify that the status of the records is **Validated**.

8. Click **Copy data to target**.

9. In the **Select a job ID to run** dialog box, in the **Run for** field, select **Criteria**, and set **Row selected by user** to **Yes**. Then click **OK**.

10. On the **Target data execution** page, click **Run**.

11. When the run is completed, refresh the page to see the latest staging status.

**Expected outcome**

- On the first try, the import succeeds for the **Customer Groups** entity and partially succeeds for the **Customers** entity.
- The **Execution summary** page shows that five records were created, three records were updated, and two records have errors.
- In the staging view, two records have errors.
- After you fix the records and run the import again, the staging view shows that all records are completed.
A composite entity is a concept that allows you to build a single entity by leveraging multiple entities that are related to each other.

What is a composite entity?

Composite entity is a concept that allows you to build a single entity by leveraging multiple entities that are related to each other. The concept is heavily used in scenarios where an entity can be represented as a single document, like Sales header/line, Invoice header/line and Vendor Catalog. This concept is applicable in asynchronous integration scenarios rather than synchronous OData scenarios, and it will only be supported from a data management platform. There is no programmatic interface for composite entities in X++. It is only supported for a data management platform that is part of XML file-based imports/exports.

Example

Sales Header and Sales Line are two different entities in the system. In case the customer requirement suggests that header and lines are part of a single document, then these two entities can be merged as a composite entity. Sample sales order entity: The composite entity (MySalesTableCompositeEntity) represents a sales orders document which is comprised of Sales Order header entity (MySalesTableEntity) and Sales Order Line entity (MySalesTableLineEntity).

Based on the linked entities, these entities can be exposed as an XML document with embedded element tags for entities. XML is the only way to expose a composite entity in data management.

```xml
<?xml version="1.0" encoding="utf-8"?>
<Document>
  <MySalesTableEntity SalesID="SO1" CurrencyCode="USD" CustAccount="Acc001">
    <MySalesLineEntity SalesPrice="2.00" QtyOrdered="10.00" LineAmount="20.00" ItemId="1000" LineNum="1.00" SalesID="SO1"/>
    <MySalesLineEntity SalesPrice="2.00" QtyOrdered="10.00" LineAmount="20.00" ItemId="4401" LineNum="2.00" SalesID="SO1"/>
  </MySalesTableEntity>
  <MySalesTableEntity SalesID="SO2" CurrencyCode="USD" CustAccount="Acc002">
    <MySalesLineEntity SalesPrice="2.00" QtyOrdered="10.00" LineAmount="20.00" ItemId="4402" LineNum="1.00" SalesID="SO2"/>
  </MySalesTableEntity>
</Document>
```
Building the composite entity

There is a node for composite data entities under Data model. Let’s take the example of MySalesTableEntity.

**Step 1: Identify and create the individual entities for the composite entity**

Make sure that the entities are related to each other. In this example the individual entities are MySalesTableEntity and MySalesLineEntity.

**Step 2: Add relations between individual entities**

Add a relation to parent entity in the relations node. Example – MySalesLineEntity has relationship to MySalesTableEntity.

**Step 3: Create a new composite entity**

1. Add a new Dynamics 365 artifact item of type Composite entity to the project.

2. In designer mode, right-click the entity and select New Root Data Entity Reference.

3. Set the data entity to parent data entity. In this case its MySalesTableEntity.

4. Right-click the parent entity node and select New Embedded Data Entity Reference.
5. Set the embedded data entity as the child entity. In this case it is MySalesLineEntity.

6. Set the **Relation** property from the drop-down list on the embedded data entity properties.

7. Composite entity supports multi-level child entities.

**Step 4: Create relationships between staging tables**
You need to create relationships between the parent and child entity staging tables based on the natural keys. For example, staging tables for MySalesTable and MySalesLine are linked by SalesID, DefinitionGroup, and ExecutionId.

1. Add a foreign key relation on MySalesLineStaging table.

2. Add two columns, RowId and ParentRowId (type int), on all the staging tables associated with the composite data entity. Refer to SysCompositeHeaderStaging table for the columns properties.
Step 5: Set up the metadata for DMFEntity

These columns are used to define runtime relationships during the target data movement.

- Create a cluster index on the staging tables which includes RowId, ParentRowId, DefinitionGroup, and ExecutionId. This is for performance reasons.
- Compile and synchronize the artifacts.

Step 6: Test the entity locally

For local testing the composite entity metadata needs to be refreshed.

1. Go to DIXF Parameters > Entity settings. Click Refresh entity list.

2. Alternately, you can write the following job to refresh the composite entity list metadata.

   ```
   DMFDataPopulation::refreshCompositeEntityList();
   ```

3. Execute the job. This refreshes the metadata required for the entity lookup.

   **NOTE**
   Currently this is a workaround. In the future a feature will be enabled to refresh the list at compile/sync time.

Step 6: Test the entity locally

We recommend that you import and export the data as a normal entity from DIXF standard process. Refer to the following the steps for importing and exporting entity.

**NOTE**

The source types of XML-Attribute or XML-Element are supported for composite entity. In entity execution parameters, composite entities cannot be imported in parallel using the parallel processing settings.

Import a composite entity

1. Click Import.
2. Enter Name, Source data format, and Entity name.
3. The Source data format is either xml-attribute or xml-element.
4. Click Import now.
5. The number of records created/updated/pending are shown.

Export a composite entity

1. Click Export.
2. Enter Name, Source data format, and Entity name.
3. Click Add entity and Export now.
4. Click Download package.

General troubleshooting guidelines

- Issue: The exported composite XML file is not imported. The scenario that produces this is:
  - Export a file for composite entity.
  - Import the same file.
  - Mapping fails and the file is not imported.
- Root cause:
  - Check if the exported file has lines or related child entity information.
  - If there no lines or related child entity information, then the lines will not be mapped during import.
- Resolution:
  - Create a sample file with all of the child entities.
  - Use this file for initial mapping only.
  - When the mapping is successful, import the actual file which does not have the line data into the entity. Use reimport or upload a new file.
  - This should import files with partial data (blank child records), depending on the validity of the records.
This procedure shows how to configure, enable, validate, and resolve conflicts when sharing data between companies. It uses the USMF company and the Financial data sharing template.

This task guide requires Dynamics AX platform 7.1 or later.

1. Go to Navigation pane > Modules > System administration > Workspaces > Data management.
2. Click Import.
3. In the Name field, type a value.
4. In the Source data format field, select the 'Package' type. Click Upload. Navigate to the location of the Financial data sharing template package file and select that file.
5. Click Save.
6. In the list, mark the selected row. Select the data sharing policy that was just created.
7. Click Import.
8. Click Close.
9. Refresh the page.
10. Close the page.
11. Close the page.
12. Close the page.
14. In the list, find and select Payment days.
15. Click Add.
16. In the list, mark the selected row.
17. In the Company field, type 'USSI'.
18. Click Add.
19. In the Company field, type 'USMF'.
20. Click Save.
21. Click Enable.
22. Click Yes.
23. Click Find sharing issues.
24. Click Yes.
25. Click Update field value.
26. Click Use value from company 1.
27. Refresh the page.
28. Close the page.
This topic demonstrates how to create a record template so that field values that are used often do not have to be entered explicitly for each new record. In this procedure, you'll create a new record for new laptops that should be added to your fixed assets. This procedure uses the USMF sample company.

1. In the navigation pane, go to Modules > Fixed assets > Fixed assets > Fixed assets.
2. Select New.
3. In the Fixed asset group field, enter or select a value.
4. In the Name field, type a value. For example, enter Corporate lead laptop.
5. In the Search name field, type a value. For example, enter laptop.
6. Expand the Technical information section.
7. In the Make, Model, and Model year fields, type values.
8. On the Action Pane, select Options.
9. Select Record info.
10. Select User template.
11. In the Name field, type a value.
12. In the Description field, type a value.
13. Select OK.
This procedure shows how to use a previously defined record template to create a new record. To complete this procedure, you must first complete the “Create a record template to facilitate data entry” procedure.

This procedure uses the USMF company.

1. In the Navigation pane, go to Fixed assets > Fixed assets > Fixed assets.
2. Click New. You will be prompted to select a template. Select the one that corresponds to your business need.
3. In the list, find and select the desired record.
4. Click OK.
The Prospect to cash solution provides direct synchronization across Dynamics 365 Supply Chain Management and Dynamics 365 Sales. The Prospect to cash templates that are available with the Data Integration feature enable the flow of data for accounts, contacts, products, sales quotations, sales orders, and sales invoices. While data is flowing, you can perform sales and marketing activities in Sales, and you can handle order fulfillment by using inventory management in Supply Chain Management.

For more information about the Prospect to cash integration, watch the short YouTube video Prospect to cash integration.

In the current version, the Prospect to cash solution provides the following types of direct synchronization:

- Synchronize accounts directly from Sales to customers in Supply Chain Management
- Synchronize products directly from Supply Chain Management to products in Sales
- Synchronize contacts directly from Sales to contacts or customers in Supply Chain Management
- Synchronize sales quotation headers and lines directly from Sales to Supply Chain Management
- Synchronization of sales orders directly between Sales and Supply Chain Management
- Synchronize sales invoice headers and lines directly from Supply Chain Management to Sales

System requirements for Supply Chain Management

Prospect to cash integration is supported on the following versions:

**Microsoft Dynamics 365 for Finance and Operations, Enterprise edition 7.3 (December 2017)**
- Dynamics 365 for Finance and Operations, Enterprise edition (December 2017) - Application build 7.3.11971.56116 with Platform Update 12 (7.0.4709.41129)

**Dynamics 365 for Finance and Operations, Enterprise edition (July 2017)**
- Dynamics 365 for Finance and Operations, Enterprise edition (July 2017) - with platform update 8 (application build 7.2.11792.56024 with platform build 7.0.4565.16212).

- The following hotfixes are required:
  - **KB4045570** – This hotfix enables sales order synchronization from Sales to Supply Chain Management via the Data Integration feature. It also provides several other enhancements.
  - **KB4036524** – This hotfix enables sales order line synchronization from Supply Chain Management to Sales via the Data Integration feature.
  - **KB4036461** – This hotfix enables sales order synchronization from Supply Chain Management to Sales via the Data Integration feature.

**NOTE**
You only have to install KB4045570 because the installation includes the changes from other hotfixes.

**Dynamics 365 for Finance and Operations version 1611 (November 2016)**
- Dynamics 365 for Finance and Operations version 1611 (November 2016) with platform update 8 or higher
• The following hotfixes are required:
  
  o **KB4051266** - Enable sales order synchronization with Data integrator from Supply Chain Management to Sales.
  
  o **KB4037542** - Enable sales order header and line synchronization with Data integrator from Supply Chain Management to Sales.
  
  o **KB4033093** - Support for prospect to cash integration through data entities is required.

**NOTE**

After you install the hotfixes, you must trigger the following batch job from the 

_SalesPopulateProspectToCash_ form. This form is hidden because you only need it once. To access the form, log in to the environment and add the following to the URL in your browser address:

&mi=action:SalesPopulateProspectToCash, for example,

https://ax123456.cloud.test.dynamics.com/?cmp=USMF&mi=action:SalesPopulateProspectToCash.

When the form opens, click OK. This will populate a new _LineCreationSequenceNumber_ field in the _SalesLine_, _SalesQuotationLine_, and _CustInvoiceTrans_ tables with unique values, and the product list will be refreshed. This is required for the Prospect to cash integration to work.

---

**System requirements for Sales**

To use the Prospect to cash solution, you must install the following components:

• Dynamics 365 Sales version 1612 (8.2.1.207) (DB 8.2.1.207) online or a later version

• Prospect to cash solution for Dynamics 365 Sales, version 1.15.0.0 or a later version. The solution is available for download from AppSource. [Download Dynamics 365, Prospect to Cash.](#)
This topic discusses the templates and underlying tasks that are used to synchronize accounts directly from Dynamics 365 Sales to Dynamics 365 Supply Chain Management.

Data flow in Prospect to cash

The Prospect to cash solution uses the Data integration feature to synchronize data across instances of Supply Chain Management and Sales. The Prospect to cash templates that are available with the Data integration feature enable the flow of data about accounts, contacts, products, sales quotations, sales orders, and sales invoices between Supply Chain Management and Sales. The following illustration shows how the data is synchronized between Supply Chain Management and Sales.

Templates and tasks

To access the available templates, open Power Apps Admin Center. Select Projects, and then, in the upper-right corner, select New project to select public templates.

The following template and underlying task are used to synchronize accounts from Sales to Supply Chain Management:

- **Name of the template in Data integration**: Accounts (Sales to Fin and Ops) - Direct
- **Name of the task in the project**: Accounts - Customers

No synchronization tasks are required before Account/Customer synchronization can occur.

Entity set
Entity flow

Accounts are managed in Sales and synchronized to Supply Chain Management as customers. The Is Externally Maintained property on these customers is set to Yes to track customers that originate from Sales. During invoicing, this information is used to filter invoices that are synchronized to Sales.

Prospect to cash solution for Sales

The Account Number column is available on the Account page. It has been made a natural and unique key in order to support the integration. The natural key feature of the Customer Relationship Management (CRM) solution might affect customers who already use the Account Number column, but who don’t use unique Account Number values per account. Currently, the integration solution doesn’t support this case.

When a new account is created, if an Account Number value doesn’t already exist, it’s automatically generated by using a number sequence. The value consists of ACC, followed by an increasing number sequence and then a suffix of six characters. Here is an example: ACC-01000-BVRCPS

When the integration solution for Sales is applied, an upgrade script sets the Account Number column for existing accounts in Sales. If there are no Account Number values, the number sequence that was mentioned earlier is used.

Preconditions and mapping setup

- The CustomerGroupId mapping must be updated to a valid value in Supply Chain Management. You can specify a default value, or you can set the value by using a value map.
  
  The default template value is 10.

- By adding the following mappings, you can help reduce the number of manual updates that are required in Supply Chain Management. You can use a default value or a value map from, for example, Country/Region or City.

  - SiteId – A site is required in order to generate quotations and sales order lines in Supply Chain Management. A default site can be taken either from the product, or from the customer from the order header.
    
    The default template value is 1.

  - WarehouseId – A warehouse is required in order to process quotations and sales order lines in Supply Chain Management. A default warehouse can be taken either from the product, or from the customer from the order header in Supply Chain Management.
    
    The default template value is 13.

  - LanguageId – A language is required in order to generate quotations and sales orders in Supply Chain Management. By default, the language from the order header from the customer is used.
    
    The default template value is en-us.

Template mapping in Data integration
NOTE

The **Payment terms**, **Freight terms**, **Delivery terms**, **Shipping method**, and **Delivery mode** columns aren't included in the default mappings. To map these columns, you must set up a value mapping that is specific to the data in the organizations that the table is synchronized between.

The following illustrations show an example of a template mapping in Data integration.

NOTE

The mapping shows which column information will be synchronized from Sales to Supply Chain Management.

![Template mapping in Data integration](image)

**Related topics**

*Prospect to cash*
Synchronize accounts directly from Sales to customers in Supply Chain Management

Synchronize contacts directly from Sales to contacts or customers in Supply Chain Management

Synchronization of sales orders directly between Sales and Supply Chain Management

Synchronize sales invoice headers and lines directly from Supply Chain Management to Sales
This topic discusses the templates and underlying tasks that are used to synchronize products directly from Dynamics 365 Supply Chain Management to Dynamics 365 Sales.

**Data flow in Prospect to cash**

The Prospect to cash solution uses the Data integration feature to synchronize data across instances of Supply Chain Management and Sales. The Prospect to cash templates that are available with the Data integration feature enable the flow of data about accounts, contacts, products, sales quotations, sales orders, and sales invoices between Supply Chain Management and Sales. The following illustration shows how the data is synchronized between Supply Chain Management and Sales.

<table>
<thead>
<tr>
<th>Sales</th>
<th>Data Integrator</th>
<th>Finance and Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account and Contact</td>
<td>Customer</td>
<td>Create party and related address information</td>
</tr>
<tr>
<td>Autonumbering</td>
<td>Contact</td>
<td>Read only based on origin</td>
</tr>
<tr>
<td>Active Customer</td>
<td>Product</td>
<td>Sellable products</td>
</tr>
<tr>
<td>(to Customer)</td>
<td>Quote</td>
<td>Autonumbering overwrite</td>
</tr>
<tr>
<td>Account relation (to Contact)</td>
<td>Quote</td>
<td>Discount transformation</td>
</tr>
<tr>
<td>Read only based on origin</td>
<td>Sales order</td>
<td>Validate customer origin to control sync of header and line</td>
</tr>
<tr>
<td>Create Price list item</td>
<td>Invoice</td>
<td>Calculate sales discount, tax and totals</td>
</tr>
<tr>
<td>Submit quote</td>
<td></td>
<td>Validate customer origin to control sync of header and line</td>
</tr>
<tr>
<td>Processing status</td>
<td></td>
<td>Calculate invoice discount, tax and totals</td>
</tr>
<tr>
<td>Read only</td>
<td></td>
<td>Read only</td>
</tr>
</tbody>
</table>

**Templates and tasks**

To access the available templates, open Power Apps Admin Center. Select **Projects**, and then, in the upper-right corner, select **New project** to select public templates.

The following template and underlying tasks are used to synchronize products from Supply Chain Management to Sales.

- **Name of the template in Data integration**: Products (Supply Chain Management to Sales) - Direct
- **Name of the task in the Data integration project**: Products

No synchronization tasks are required before product synchronization can occur.

**Entity set**
Sellable released products | Products

**Entity flow**

Products are managed in Supply Chain Management and synchronized to Sales. The **Sellable released products** data entity in Supply Chain Management exports only products that are **sellable**. Sellable products are products that have the information that they require in order to be used on a sales order. The same rules apply when a product is validated by using the **Validate** function on the **Released product** page.

The product number is used as a key. Therefore, when product variants are synchronized to Sales, each product variant has an individual product ID.

**Prospect to cash solution for Sales**

In Sales, a new **Is Externally Maintained** field has been added on products to indicate that a given product is maintained externally. By default, the value is set to **Yes** during an import to Sales. The following values are available:

- **Yes** – The product originated from Supply Chain Management and won't be editable in Sales.
- **No** – The product was entered directly in Sales.
- **(Blank)** – The product existed in Sales before the Prospect to cash solution was enabled.

The **Is Externally Maintained** field helps ensure that only quotations and sales orders that have externally maintained products will be synchronized to Supply Chain Management.

Externally maintained products are automatically added to the first valid price list that has the same currency. Price lists are organized alphabetically by name. The product sales price from Supply Chain Management is used as the price on the price list. Therefore, there must be a price list in Sales for every product sales currency in Supply Chain Management. The currency on the released sellable products is set to the accounting currency in the legal entity that the product is exported from.

**NOTE**

- Product synchronization will not succeed unless there is a price list that has a matching currency.
- You can control the used price list with the integration by mapping the pricelevelid.name [Default Price List (Name)] in the Data Integration project. The input has to be in all lowercase letters. For example, the default for a price list in Sales named ‘Standard’ would be: Destination field: pricelevelid.name [Default Price List (Name)] and Map type: [ { "transformType": "Default", "defaultValue": "standard" } ].

**Preconditions and mapping setup**

- Before you run the synchronization for the first time, you must fill the Distinct product table for existing products in Supply Chain Management. Existing products won't be synchronized until this job is completed.
  1. In Supply Chain Management, use Search to search for **Populate distinct product table**.
  2. Select **Populate distinct product table** to run the job. This job must be run only one time.
- Make sure that the required value map for the selling unit of measure (UOM) between Supply Chain Management and Sales exists in the mapping of **SalesUnitSymbol** to **DefaultUnit (Name)**.
- Update the value map for **Unit group (defaultuomscheduleid.name)** so that it matches **Unit groups** in Sales.
The default template value is **Default unit**.

- Make sure that the selling UOMs for all products from Supply Chain Management exist in Sales.
- Make sure that price lists exist in Sales for every product sales currency in Supply Chain Management.
- When products are created in Sales, they can have a status of **Draft** or **Active**. The behavior is controlled at **Settings > Administration > System settings > Sales** in Sales.

  Products that have **Draft** status when they are created must be activated before they can be added to quotations or sales orders.

### Template mapping in Data integration

The following illustration shows an example of a template mapping in Data integration.

**NOTE**

The mapping shows which field information will be synchronized from Sales to Supply Chain Management.

![Template mapping illustration](image)

**Related topics**

- **Prospect to cash**
  - Synchronize accounts directly from Sales to customers in Supply Chain Management
  - Synchronize contacts directly from Sales to contacts or customers in Supply Chain Management
  - Synchronization of sales orders directly between Sales and Supply Chain Management
Synchronize sales invoice headers and lines directly from Supply Chain Management to Sales
This topic discusses the templates and underlying tasks that are used to synchronize Contact (Contacts) and Contact (Customers) tables directly from Dynamics 365 Sales to Dynamics 365 Supply Chain Management.

Data flow in Prospect to cash

The Prospect to cash solution uses the Data integration feature to synchronize data across instances of Supply Chain Management and Sales. The Prospect to cash templates that are available with the Data integration feature enable the flow of data about accounts, contacts, products, sales quotations, sales orders, and sales invoices between Supply Chain Management and Sales. The following illustration shows how the data is synchronized between Supply Chain Management and Sales.

Templates and tasks

To access the available templates, open PowerApps Admin Center. Select Projects, and then, in the upper-right corner, select New project to select public templates.

The following templates and underlying tasks are used to synchronize Contact (Contacts) tables in Sales to Contact (Customers) tables in Supply Chain Management.

- **Names of the templates in Data integration**
  - Contacts (Sales to Supply Chain Management) - Direct
  - Contacts to Customer (Sales to Supply Chain Management) - Direct

- **Names of the tasks in the Data integration project**
  - Contacts
The following synchronization task is required before contact synchronization can occur: Accounts (Sales to Supply Chain Management)

### Entity sets

<table>
<thead>
<tr>
<th>SALES</th>
<th>SUPPLY CHAIN MANAGEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contacts</td>
<td>Dataverse Contacts</td>
</tr>
<tr>
<td>Contacts</td>
<td>Customers V2</td>
</tr>
</tbody>
</table>

### Entity flow

Contacts are managed in Sales and synchronized to Supply Chain Management.

A contact in Sales can become either a contact or a customer in Supply Chain Management. To determine whether a contact in Sales should be synchronized to Supply Chain Management as a contact or a customer, the system looks at the following properties on the contact in Sales:

- **Synchronization to a customer in Supply Chain Management**: Contacts where Is Active Customer is set to Yes
- **Synchronization to a contact in Supply Chain Management**: Contacts where Is Active Customer is set to No and Company (parent account/contact) points to an account (not a contact)

### Prospect to cash solution for Sales

A new **Is Active Customer** column has been added to the contact. This column is used to differentiate contacts that have sales activity and contacts that don’t have sales activity. **Is Active Customer** is set to Yes only for contacts that have related quotations, orders, or invoices. Only those contacts are synchronized to Supply Chain Management as customers.

A new **IsCompanyAnAccount** column has been added to the contact. This column indicates whether a contact is linked to a company (parent account/contact) of the Account type. This information is used to identify contacts that should be synchronized to Supply Chain Management as contacts.

A new **Contact Number** column has been added to the contact to help guarantee a natural and unique key for the integration. When a new contact is created, a **Contact Number** value is automatically generated by using a number sequence. The value consists of CON, followed by an increasing number sequence and then a suffix of six characters. Here is an example: CON-01000-BVRCPS

When the integration solution for Sales is applied, an upgrade script sets the **Contact Number** column for existing contacts by using the number sequence that was mentioned earlier. The upgrade script also sets the **Is Active Customer** column to Yes for any contacts that have sales activity.

### In Supply Chain Management

Contacts are tagged by using the **IsContactPersonExternallyMaintained** property. This property indicates that a given contact is maintained externally. In this case, externally maintained contacts are maintained in Sales.

### Preconditions and mapping setup

**Contact to customer**
- **CustomerGroup** is required in Supply Chain Management. To help prevent synchronization errors, you can specify a default value in the mapping. That default value is then used if the column is left blank in Sales.

  The default template value is 10.

- By adding the following mappings, you can help reduce the number of manual updates that are required in Supply Chain Management. You can use a default value or a value map from, for example, **Country/Region** or **City**.
  - **SiteId** – A default site can also be defined on products in Supply Chain Management. A site is required in order to generate quotations and sales orders in Supply Chain Management.
    
    A template value for **SiteId** isn't defined.
  
  - **Warehouseld** – A default warehouse can also be defined on products in Supply Chain Management. A warehouse is required in order to generate quotations and sales orders in Supply Chain Management.
    
    A template value for **Warehouseld** isn't defined.
  
  - **LanguageId** – A language is required in order to generate quotations and sales orders in Supply Chain Management.
    
    The default template value for **LanguageId** is **en-us**.

**Template mapping in Data integration**

The following illustrations show an example of a template mapping in Data integration.

**NOTE**

The mapping shows which column information will be synchronized from Sales to Supply Chain Management.

**Contact to contact example**
Contact to customer example
### Related topics

**Prospect to cash**

Synchronize accounts directly from Sales to customers in Supply Chain Management

Synchronize products directly from Supply Chain Management to products in Sales

Synchronization of sales orders directly between Sales and Supply Chain Management

Synchronize sales invoice headers and lines directly from Supply Chain Management to Sales
The topic discusses the templates and underlying tasks that are used to synchronize sales quotation headers and lines directly from Dynamics 365 Sales to Dynamics 365 Supply Chain Management.

**NOTE**
Before you can use the Prospect to cash solution, you should be familiar with [Integrate data into Microsoft Dataverse for Apps](#).

### Data flow in Prospect to cash

The Prospect to cash solution uses the Data integration feature to synchronize data across instances of Supply Chain Management and Sales. The Prospect to cash templates that are available with the Data integration feature enable the flow of data for accounts, contacts, products, sales quotations, sales orders, and sales invoices between Supply Chain Management and Sales. The following illustration shows how the data is synchronized between Supply Chain Management and Sales.

The following template and underlying tasks are used to synchronize sales quotation headers and lines directly from Sales to Supply Chain Management:

- **Name of the template in Data integration**: Sales Quotes (Sales to Supply Chain Management) - Direct
- **Names of the tasks in the Data integration project**:
  - QuoteHeader
  - QuoteLine

The following synchronization tasks are required before synchronization of sales quotation headers and lines can occur:
### Entity set

<table>
<thead>
<tr>
<th>SALES</th>
<th>SUPPLY CHAIN MANAGEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quotes</td>
<td>Dataverse sales quotation header</td>
</tr>
<tr>
<td>QuoteDetails</td>
<td>Dataverse sales quotation lines</td>
</tr>
</tbody>
</table>

### Entity flow

Sales quotations are created in Sales and synchronized to Supply Chain Management.

Sales quotations from Sales are synchronized only if the following conditions are met:

- All quote products on the sales quotation are externally maintained.
- After you click **Activate quote**, the sales quotation is active.

### Prospect to cash solution for Sales

The **Has Externally Maintained Products Only** field has been added to the **Quote** entity to consistently track whether the sales quotation consists entirely of externally maintained products. If a sales quotation has only externally maintained products, the products are maintained in Supply Chain Management. This behavior helps guarantee that you don't try to synchronize sales quotation lines that have products that are unknown to Supply Chain Management.

All quote products on the sales quotation are updated with the **Has Externally Maintained Products Only** information from the sales quotation header. This information is found in the **Quote Has Externally Maintained Products Only** field on the **QuoteDetails** entity.

A discount can be added to the quote product and will be synchronized to Supply Chain Management. The **Discount**, **Charges**, and **Tax** fields on the header are controlled by a setup in Supply Chain Management. Currently, this setup doesn't support integration mapping. In the current design, the **Price**, **Discount**, **Charge**, and **Tax** fields are maintained and handled in FSupply Chain Management.

In Sales, the solution makes the following fields read-only, because the values aren't synchronized to Supply Chain Management:

- Read-only fields on the sales quotation header: **Discount %**, **Discount**, and **Freight Amount**
- Read-only fields on quote products: **Tax**

### Preconditions and mapping setup

Before sales quotations are synchronized, it's important that you update the following settings.

**Setup in Sales**

- Make sure that permissions are set up for the team that the user from your connection set in Sales is assigned to. If you're using demo data, the user usually has admin access, but the team doesn't have admin access. If the team doesn't have admin access when you run the project from Data integration, you will receive an error message that states that the Principal team is missing.

To set up permissions for the team, go to **Settings > Security > Teams**, and select the relevant team.
Select **Manage Roles**, and then select a role that has the desired permissions, such as **System Administrator**.

- Go to **Settings > Administration > System settings > Sales**, and make sure that the following settings are used:
  - The **Use system prizing calculation system** option is set to **Yes**.
  - The **Discount calculation method** field is set to **Line item**.

**Setup in the Data integration project**

**QuoteHeader**

- Make sure that the required mapping exists for **Shipto_country** to **DeliveryAddressCountryRegionISOCode**. In the value map, you can define a default value that is used if the value is left blank. Just leave the left side blank, and set the right side to the desired country or region. In this way, you don't have to type the country or region for national orders.

  The template value is a value map where several countries or regions are mapped, and where a blank value equals a value of **US**.

**QuoteLine**

- Make sure that the required value map exists for **SalesUnitSymbol** in Supply Chain Management.

- Make sure that the required units are defined in Sales.

  A template value that has a value map is defined for **oumid.name** to **SalesUnitSymbol**.

- Optional: You can add the following mappings to help guarantee that sales quotation lines are imported into Supply Chain Management if there is no default information from either the customer or the product:
  - **SiteId** – A site is required in order to generate quotations and sales order lines in Supply Chain Management. There is no default template value for **SiteId**.
  - **WarehouseId** – A warehouse is required in order to process quotations and sales order lines in Supply Chain Management. There is no default template value for **WarehouseId**.

**Template mapping in data integrator**

**NOTE**

- The **Discount**, **Charges**, and **Tax** fields are controlled by a complex setup in Supply Chain Management. Currently, this setup doesn't support integration mapping. In the current design, the **Price**, **Discount**, **Charge**, and **Tax** fields are handled by Supply Chain Management.

- The **Payment terms**, **Freight terms**, **Delivery terms**, **Shipping method**, and **Delivery mode** fields aren't part of the default mappings. To map these fields, you must set up a value mapping that is specific to the data in the organizations that the entity is synchronized between.

The following illustrations show an example of a template mapping in data integrator.

**QuoteHeader**

- **S...
### QuoteLine

**Source:** Sales.quote
d**Destination:** Fin and Ope.CDS sales quotation lines

<table>
<thead>
<tr>
<th>Source Field</th>
<th>Map Type</th>
<th>Destination Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>productid.productnumber [Existing Product (Product ID)]</td>
<td>=</td>
<td>ProductNumber [ProductNumber]</td>
</tr>
<tr>
<td>quantity [Quantity]</td>
<td>=</td>
<td>RequestedSalesQuantity [RequestedSalesQuantity]</td>
</tr>
<tr>
<td>quoted quotations [Quote (Quote ID)]</td>
<td>=</td>
<td>SalesQuotationNumber [SalesQuotationNumber]</td>
</tr>
<tr>
<td>sequencenumber [Sequence Number]</td>
<td>=</td>
<td>LineCreationSequenceNumber [LineCreationSequenceNumber]</td>
</tr>
<tr>
<td>unitname [Unit (Name)]</td>
<td>Fn</td>
<td>SalesUnitSymbol [SalesUnitSymbol]</td>
</tr>
<tr>
<td>manualdiscountamount [Manual Discount]</td>
<td>=</td>
<td>TotalDiscountAmount [TotalDiscountAmount]</td>
</tr>
<tr>
<td>priceperunit [Price Per Unit]</td>
<td>=</td>
<td>SalesPrice [SalesPrice]</td>
</tr>
</tbody>
</table>

**Related topics**

**Prospect to cash**
The topic discusses the templates and underlying tasks that are used to run synchronization of sales orders directly between Dynamics 365 Sales and Dynamics 365 Supply Chain Management.

**Data flow in Prospect to cash**

The Prospect to cash solution uses the Data integration feature to synchronize data across instances of Supply Chain Management and Sales. The Prospect to cash templates that are available with the Data integration feature enable the flow of data for accounts, contacts, products, sales quotations, sales orders, and sales invoices between Supply Chain Management and Sales. The following illustration shows how the data is synchronized between Supply Chain Management and Sales.

To access the available templates, open Power Apps Admin Center. Select Projects, and then, in the upper-right corner, select New project to select public templates.

The following templates and underlying tasks are used to run synchronization of sales orders directly between Sales and Supply Chain Management.

- **Names of the templates in Data integration:**
  - Sales Orders (Sales to Supply Chain Management) - Direct
  - Sales Orders (Supply Chain Management to Sales) - Direct

- **Names of the tasks in the Data integration project:**
  - OrderHeader
  - OrderLine

The following synchronization tasks are required before synchronization of sales invoice headers and lines can occur:

- Products (Supply Chain Management to Sales) - Direct
Entity set

<table>
<thead>
<tr>
<th>SUPPLY CHAIN MANAGEMENT</th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataverse sales order headers</td>
<td>SalesOrders</td>
</tr>
<tr>
<td>Dataverse sales order lines</td>
<td>SalesOrderDetails</td>
</tr>
</tbody>
</table>

Entity flow

Sales orders are created in Sales and synchronized to Supply Chain Management when Run project is triggered for a project based on the Sales Orders (Sales to Supply Chain Management) - Direct template. You can only activate and synchronize orders from Sales if all Order Products consist of products that are externally maintained. Therefore, there can be no write-in products. After the order is activated, the sales order becomes read-only in the user interface (UI). At that point, the updates are made from Supply Chain Management. After the sales order has a status of Confirmed, the a project based on the Sales Orders (Supply Chain Management to Sales) - Direct template can be used to synchronize updates or fulfillment status from Supply Chain Management to Sales.

You don’t have to create orders in Sales. You can create new sales orders in Supply Chain Management instead. After a sales order has a status of Confirmed, it’s synchronized to Sales as described in the previous paragraph.

In Supply Chain Management, filters in the template help guarantee that only the relevant sales orders are included in the synchronization:

- On the sales order, both the ordering customer and the invoicing customer have to originate from Sales to be included in the synchronization. In Supply Chain Management, the OrderingCustomerIsExternallyMaintained and InvoiceCustomerIsExternallyMaintained columns are used to filter sales orders from the data tables.
- The sales order in Supply Chain Management must be confirmed. Only confirmed sales orders or sales orders that have a higher processing status, such as Shipped or Invoiced, are synchronized to Sales.
- After a sales order is created or modified, the Calculate sales totals batch job in Supply Chain Management must be run. Only sales orders where sales totals are calculated will be synchronized to Sales.

Freight tax

Sales doesn’t support tax at the header level, because tax is stored at the line level. To support tax at the header level from Supply Chain Management, such as tax on freight, the system synchronizes tax to Sales as a write-in product that is named Freight Tax, and that has the tax amount from Supply Chain Management. In this way, the standard price calculation in Sales can be used for totals, even when there is tax at the header level from Supply Chain Management.

Discount calculation and rounding

The discount calculation model in Sales differs from the discount calculation model in Supply Chain Management. In Supply Chain Management, the final discount amount on a sales line can be the result of a combination of discount amounts and discount percentages. If this final discount amount is divided by the quantity on the line, rounding can occur. However, this rounding isn’t considered if a rounded per-unit discount amount is synchronized to Sales. To help guarantee that the full discount amount from a sales line in Supply Chain Management is correctly synchronized to Sales, the full amount must be synchronized without being
Prospect to cash solution for Sales
divided by the line quantity. Therefore, you must define the **Discount calculation method** as **Line item** in Sales.

When a sales order line is synchronized from Sales to Supply Chain Management, the full line discount amount is used. Because Supply Chain Management has no field that can store the full discount amount for a line, the amount is divided by the quantity and stored in the **Line discount** field. Any rounding that occurs in this division is stored in the **Sales charges** field on the sales line.

**Example**

**Synchronization from Sales to Supply Chain Management**

- **Sales**: Quantity = 3, per-line discount = $10.00
- **Supply Chain Management**: Quantity = 3, line discount amount = $3.33, sales charge = -$0.01

**Synchronization from Supply Chain Management to Sales**

- **Supply Chain Management**: Quantity = 3, line discount amount = $3.33, sales charge = -$0.01
- **Sales**: Quantity = 3, per-line discount = (3 × $3.33) + $0.01 = $10.00

**Prospect to cash solution for Sales**

New columns have been added to the **Order** table and appear on the page:

- **Is Maintained Externally** – Set this option to **Yes** when the order is coming from Supply Chain Management.
- **Processing status** – This column shows the processing status of the order in Supply Chain Management. The following values are available:
  - **Draft** – The initial status when an order is created in Sales. In Sales, only orders with this processing status can be edited.
  - **Active** – The status after the order is activated in Sales by using the **Activate** button.
  - **Confirmed**
  - **Packing Slip**
  - **Invoiced**
  - **Picked**
  - **Partially Picked**
  - **Partially Packed**
  - **Shipped**
  - **Partially Shipped**
  - **Partially Invoiced**
  - **Cancelled**

The **Has Externally Maintained Products Only** setting is used during order activation to consistently track whether a sales order consists entirely of externally maintained products. If a sales order consists entirely of externally maintained products, the products are maintained in Supply Chain Management. This setting helps guarantee that you don't activate and try to synchronize sales order lines that have products that are unknown to Supply Chain Management.

The **Create Invoice**, **Cancel Order**, **Recalculate**, **Get Products**, and **Lookup Address** buttons on the **Sales order** page are hidden for externally maintained orders, because invoices will be created in Supply Chain Management and synchronized to Sales. These orders can't be edited, because sales order information will be synchronized from Supply Chain Management after activation.

The sales order status will remain **Active** to help guarantee that changes from Supply Chain Management can
flow to the sales order in Sales. To control this behavior, set the default Statecode [Status] value to Active in the Data integration project.

Preconditions and mapping setup

Before you synchronize sales orders, it's important that you update the following settings in the systems.

Setup in Sales

- Make sure that permissions are set up for the team that the user from your Sales connection set is assigned to. If you're using demo data, the user usually has admin access, but the team doesn't have admin access. If the team doesn't have admin access, when you run the project from Data integration, you will receive an error message that states that the Principal team is missing.

  Go to Settings > Security > Teams, select the relevant team, select Manage Roles, and select a role that has the desired permissions, such as System Administrator.

- To ensure correct calculation of discounts in both Sales and Supply Chain Management Discount calculation method must be set to Line item.

  Go to Settings > Administration > System settings > Sales, and make sure that the following settings are used:

  - The Use system prizing calculation system option is set to Yes.
  - The Discount calculation method column is set to Line item.

Setup in Supply Chain Management

- Go to Sales and marketing > Periodic tasks > Calculate sales totals, and set the job to run as a batch job. Set the Calculate totals for sales orders option to Yes. This step is important, because only sales orders where sales totals are calculated will be synchronized to Sales. The frequency of the batch job should be aligned with the frequency of sales order synchronization.

If you also use work order integration, you need to set up the sales origin. The sales origin is used to distinguish sales orders in Supply Chain Management that were created from work orders in Field Service. When a sales order has a sales origin of the Work order integration type, the External work order status field appears on the sales order header. Additionally, the sales origin ensures that sales orders that were created from work orders in Field Service are filtered out during sales order synchronization from Supply Chain Management to Field Service.

1. Go to Sales and marketing > Setup > Sales orders > Sales origin.
2. Select New to create a new sales origin.
3. In the Sales origin column, enter a name for the sales origin, such as SalesOrder.
4. In the Description column, enter a description, such as Sales Order from Sales.
5. Select the Origin type assignment check box.
6. Set the Sales origin type column to Sales order integration.
7. Select Save.

Setup in the Sales Orders (Sales to Supply Chain Management) - Direct Data integration project

- Make sure that the required mapping exists for Shipto_country to DeliveryAddressCountryRegionISOCode. You can make blank a default value in the value map to avoid having to type country for national orders. Set the left side to 'Blank', and set the right side to the desired country or region.

  The template value is a value map where several countries or regions are mapped, and where 'Blank' = US.

Setup in the Sales Orders (Supply Chain Management to Sales) - Direct Data integration project
SalesHeader task

- A price list is required in order to create orders in Sales. Update the value map for `pricelevelid.name [Price List Name]` to the price list that is used in Sales per currency. You can use the default price list for a single currency. Alternatively, if you have price lists in multiple currencies, you can use a value map.

  The default template value for `pricelevelid.name [Price List Name]` is CRM Service USA (sample).

SalesLine task

- Make sure that the required value map for `SalesUnitSymbol` in Supply Chain Management exists.

- Make sure that the required units are defined in Sales.

  A template value that has a value map is defined for `SalesUnitSymbol` to `oumid.name`.

Template mapping in Data integration

**NOTE**

The Payment terms, Freight terms, Delivery terms, Shipping method, and Delivery mode columns aren’t part of the default mappings. To map these columns, you must set up a value mapping that is specific to the data in the organizations that the table is synchronized between.

The following illustrations show an example of a template mapping in Data integration.

**NOTE**

The mapping shows which column information will be synchronized from Sales to Supply Chain Management, or from Supply Chain Management to Sales.

**Sales Orders (Supply Chain Management to Sales) - Direct: OrderHeader**
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales Orders (Supply Chain Management to Sales) - Direct: OrderLine</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Sales Orders (Sales to Supply Chain Management) - Direct: OrderHeader

**Source**
SalesOrderHeaders

**Destination**
Fin and Op CoDS sales order headers

<table>
<thead>
<tr>
<th>Source Field</th>
<th>Map Type</th>
<th>Destination Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustomerId.AccountNumber</td>
<td>=</td>
<td>Ordering.CustomerAccountNumber [Ordering.CustomerAccountNumber]</td>
</tr>
<tr>
<td>ship_to_city</td>
<td>=</td>
<td>DeliveryAddressCity [DeliveryAddressCity]</td>
</tr>
<tr>
<td>ship_to_country [Ship To Country/Region]</td>
<td>fn</td>
<td>DeliveryAddressCountryRegionISOCode [DeliveryAddressCountryRegionISOCode]</td>
</tr>
<tr>
<td>transactioncurrency.id.currencyCode</td>
<td>=</td>
<td>CurrencyCode [CurrencyCode]</td>
</tr>
<tr>
<td>ship_to_line1 [Ship To Street 1]</td>
<td>=</td>
<td>DeliveryAddressStreet [DeliveryAddressStreet]</td>
</tr>
<tr>
<td>ship_to_postalcode (Ship To ZIP/Postal Code)</td>
<td>=</td>
<td>DeliveryAddressZipCode [DeliveryAddressZipCode]</td>
</tr>
<tr>
<td>orderNumber [Order ID]</td>
<td>=</td>
<td>SalesOrderNumber [SalesOrderNumber]</td>
</tr>
<tr>
<td>ship_to_line2 [Ship To Street 2]</td>
<td>=</td>
<td>DeliveryAddressStateId [DeliveryAddressStateId]</td>
</tr>
<tr>
<td>ship_to_stateprovince [Ship To State/Province]</td>
<td>=</td>
<td>DeliveryAddressDescription [DeliveryAddressDescription]</td>
</tr>
<tr>
<td>totalAmount [Total Amount]</td>
<td>=</td>
<td>OrderTotalAmount [OrderTotalAmount]</td>
</tr>
<tr>
<td>name [Name]</td>
<td>=</td>
<td>CustomersOrderReference [CustomersOrderReference]</td>
</tr>
</tbody>
</table>

None  

None  

None
## Sales Orders (Sales to Supply Chain Management) - Direct: OrderLine

<table>
<thead>
<tr>
<th>SOURCE FIELD</th>
<th>MAP TYPE</th>
<th>DESTINATION FIELD</th>
</tr>
</thead>
<tbody>
<tr>
<td>productid.productnumber (Existing Product (Product ID))</td>
<td>=</td>
<td>ProductNumber (ProductNumber)</td>
</tr>
<tr>
<td>quantity (Quantity)</td>
<td>=</td>
<td>OrderedSalesQuantity (OrderedSalesQuantity)</td>
</tr>
<tr>
<td>salesorderid.orderid (Order (Order ID))</td>
<td>=</td>
<td>SalesOrderNumber (SalesOrderNumber)</td>
</tr>
<tr>
<td>sequence (Sequence Number)</td>
<td>=</td>
<td>LineCreationSequenceNumber (LineCreationSequenceNumber)</td>
</tr>
<tr>
<td>uid.id.name (Unit (Name))</td>
<td>Fr</td>
<td>UnitSymbol (UnitSymbol)</td>
</tr>
<tr>
<td>manualdiscountamount (Manual Discount)</td>
<td>=</td>
<td>TotalDiscountAmount (TotalDiscountAmount)</td>
</tr>
<tr>
<td>priceperunit (Price Per UNIT)</td>
<td>=</td>
<td>SalesPrice (SalesPrice)</td>
</tr>
</tbody>
</table>

### Related topics

Prospect to cash
Synchronize sales invoice headers and lines directly from Finance and Operations to Sales

11/24/2021 • 3 minutes to read • Edit Online

This topic discusses the templates and underlying tasks that are used to synchronize sales invoice headers and lines directly from Dynamics 365 Supply Chain Management to Dynamics 365 Sales.

Data flow in Prospect to cash

The Prospect to cash solution uses the Data integration feature to synchronize data across instances of Supply Chain Management and Sales. The Prospect to cash templates that are available with the Data integration feature enable the flow of data about accounts, contacts, products, sales quotations, sales orders, and sales invoices between Supply Chain Management and Sales. The following illustration shows how the data is synchronized between Supply Chain Management and Sales.

Templates and tasks

To access the available templates, open Power Apps Admin Center. Select Projects, and then, in the upper-right corner, select New project to select public templates.

The following template and underlying tasks are used to synchronize sales invoice headers and lines from Supply Chain Management to Sales:

- **Name of the template in Data integration:** Sales Invoices (Fin and Ops to Sales) - Direct
- **Names of the tasks in the Data integration project:**
  - SalesInvoiceHeader
  - SalesInvoiceLine

The following synchronization tasks are required before synchronization of sales invoice headers and lines can occur:

- Products (Supply Chain Management to Sales) - Direct
- Accounts (Sales to Supply Chain Management) - Direct (if used)
- Contacts (Sales to Supply Chain Management) - Direct (if used)
Entity set

<table>
<thead>
<tr>
<th>SUPPLY CHAIN MANAGEMENT</th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Externally maintained customer sales invoice headers</td>
<td>Invoices</td>
</tr>
<tr>
<td>Externally maintained customer sales invoice lines</td>
<td>InvoiceDetails</td>
</tr>
</tbody>
</table>

Entity flow

Sales invoices are created in Supply Chain Management and synchronized to Sales.

**NOTE**

Currently, tax that is related to charges on the sales invoice header isn't included in the synchronization from Supply Chain Managements to Sales. Sales doesn't support tax information at the header level. However, tax that is related to charges at the line level is included in the synchronization.

Prospect to cash solution for Sales

- An **Invoice number** field has been added to the **Invoice** entity and appears on the page.
- The **Create invoice** button on the **Sales order** page is hidden, because invoices will be created in Supply Chain Management and synchronized to Sales. The **Invoice** page can't be edited, because invoices will be synchronized from Supply Chain Management.
- The **Sales order status** value is automatically changed to **Invoiced** when the related invoice from Supply Chain Management has been synchronized to Sales. Additionally, the owner of the sales order that the invoice was created from is assigned as the owner of the invoice. Therefore, the owner of the sales order can view the invoice.

Preconditions and mapping setup

Before you synchronize sales invoices, it's important that you update the following settings in the systems.

**Setup in Sales**

Go to **Settings > Administration > System settings > Sales**, and make sure that the following settings are used:

- The **Use system prizing calculation system** option is set to **Yes**.
- The **Discount calculation method** field is set to **Line item**.

**Setup in the Data integration project**

**SalesInvoiceHeader task**

- Make sure that the required mapping exists for **InvoiceCountryRegionId** to **BillingAddress_Country**.
  
  The template value is a value map where several countries or regions are mapped.

- A price list is required in order to create invoices in Sales. Update the value map for **pricelevelid.name** to the price list that is used in Sales per currency. You can use the default price list for a single currency. Alternatively, if you have price lists in multiple currencies, you can use a value map.

  The template value for **pricelevelid.name** is a value map that is based on currency with USD = CRM Service USA (sample).
**Template mapping in Data integration**

**NOTE**

The Payment terms, Freight terms, Delivery terms, Shipping method, and Delivery mode fields aren't included in the default mappings. To map these fields, you must set up a value mapping that is specific to the data in the organizations that the entity is synchronized between.

The following illustrations show an example of a template mapping in Data integration.

**NOTE**

The mapping shows which field information will be synchronized from Sales to Supply Chain Management.

![SalesInvoiceHeader](image)

**SalesInvoiceLine**
Related topics

**Prospect to cash**

Synchronize accounts directly from Sales to customers in Supply Chain Management

Synchronize products directly from Supply Chain Management to products in Sales

Synchronize contacts directly from Sales to contacts or customers in Supply Chain Management

Synchronization of sales orders directly between Sales and Supply Chain Management
Supply Chain Management enables synchronization of business processes between Dynamics 365 Supply Chain Management and Dynamics 365 Field Service. The integration scenarios are configured by using extensible Data integrator templates and Microsoft Dataverse to enable the synchronization of business processes. Standard templates can be used to create custom integration projects, where additional standard and custom columns and tables can be mapped to adjust the integration and meet specific business needs.

The field service integration builds on top of the existing prospect-to-cash functionality.

The first phase of the integration between Field Service and Supply Chain Management is focused on enabling work orders and agreements in Field Service to be invoiced in Supply Chain Management. The supported flow starts in Field Service, where information from work orders is synchronized to Supply Chain Management as sales orders. In Supply Chain Management, the sales orders are invoiced to generate invoice documents. In addition, the information from Field Service agreement invoices is synchronized to Supply Chain Management.

The first phase of the integration between Field Service and Supply Chain Management enables synchronization of the following items:

- Synchronize products in Supply Chain Management to products in Field Service
- Synchronize work orders in Field Service to sales orders in Supply Chain Management
- Synchronize agreement invoices in Field Service to free text invoices in Supply Chain Management

To see an example of how you can synchronize a work order between Field Service and Supply Chain Management, watch the short YouTube video How to synchronize a work order with Microsoft Dynamics 365 Integration.
Integration with Field Service, including inventory and project information

The additional functionality in this second phase focused on giving field technicians insight about the inventory information from Supply Chain Management, allowing them to update inventory levels and do material transfers. In addition, companies installing or servicing sold goods will benefit from better control and visibility to the full sales and service process with integration from projects.

**Functionality includes integration of:**

- Warehouse information
- On-hand inventory information
- Inventory transfers
- Inventory adjustments
- Supply Chain Management projects connected with Dynamics 365 Field Service work orders
- Dynamics 365 Field Service work orders with link to Supply Chain Management projects, apply this project number to the sales order to allow invoicing from the project.

The second phase of the integration between Field Service and Supply Chain Management enables synchronization with the following templates:

- Warehouses (Supply Chain Management to Field Service) - Warehouses from Supply Chain Management to Field Service [Advanced Query]
- Product Inventory (Supply Chain Management to Field Service) - Inventory level information from Supply Chain Management to Field Service [Advanced Query]
- Inventory Adjustment (Field Service to Supply Chain Management) - Inventory adjustments from Field Service to Supply Chain Management [Advanced Query]
- Inventory Transfers (Field Service to Supply Chain Management) - Inventory transfers from Field Service to Supply Chain Management [Advanced Query]
- Projects (Supply Chain Management to Field Service) - Project list from Supply Chain Management to Field Service
- Work Orders with Project (Field Service to Supply Chain Management) - Work orders in Field Service to Sales orders in Supply Chain Management, with support for Project [Advanced Query]
- Field Service Products with Inventory unit (Supply Chain Management to Sales) - Supply Chain Management ‘Sellable released products’ to Sales ‘Products’ for Field Service, including Inventory unit

**System requirements**

*System requirements for Supply Chain Management*
Field Service integration supports the following versions:

- Dynamics 365 for Finance and Operations version 8.1.2 (December 2018) was released in December 2018 and has an application build number 8.1.195 with Platform update 22 (7.0.5095).

**System requirements for Field Service**

To use the Field Service integration solution, you must install the following components:

- Field Service (version 8.2.0.286) or a later version on Dynamics 365 9.1.x - Released November 2018
- Prospect to Cash (P2C) solution for Dynamics 365, version 1.15.0.1 or a later version. The solution is available for download from AppSource.
- 'Field Service Integration, Project and Inventory' solution for Dynamics 365, version 2.0.0.0 or a later version. The solution is available for download from AppSource.
This topic discusses the templates and underlying task that are used to synchronize products from Dynamics 365 Supply Chain Management to Dynamics 365 Field Service.

The used Field Service Products (Supply Chain Management to Field Service) template is based on the Products (Supply Chain Management to Sales) – Direct template from Prospect to Cash. For more information, see Products (Supply Chain Management to Sales) – Direct.

This topic only describes the differences between the Field Service Products (Supply Chain Management to Field Service) and Products (Supply Chain Management to Sales) – Direct templates.

Templates and tasks

Name of the template in Data integration
- Field Service Products (Supply Chain Management to Field Service)

Name of the task in the Data integration project
- Products - Products

The Field Service Products (Supply Chain Management to Field Service) template includes one mapping that isn't included in the Products (Supply Chain Management to Sales) – Direct template. This mapping ensures that the required Field Service-specific field Service Product Type is set correctly.

<table>
<thead>
<tr>
<th>FIELDSERVICEPRODUCTTYPE</th>
<th>Fn</th>
<th>msdyn_fieldserviceproducttype</th>
</tr>
</thead>
<tbody>
<tr>
<td>inventory</td>
<td>690970000</td>
<td></td>
</tr>
<tr>
<td>nonInventory</td>
<td>690970001</td>
<td></td>
</tr>
<tr>
<td>service</td>
<td>690970002</td>
<td></td>
</tr>
</tbody>
</table>

The following value mapping is used.

In Supply Chain Management, the Field Service product type value on the Sellable released products data entity is calculated as follows:

- **Inventory**: Product type = Product and Item model group, Stocked product = True
- **NonInventory**: Product type = Product and Item model group, Stocked product = False
- **Service**: Product type = Service

Template mapping in Data integration

The following illustrations show the template mapping in Data integration.

Field Service Products (Supply Chain Management to Field Service): Products - Products
<table>
<thead>
<tr>
<th>Source Field</th>
<th>Map Type</th>
<th>Destination Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrencyCode</td>
<td></td>
<td>transactioncurrencyid:currencycode</td>
</tr>
<tr>
<td>ProductDescription</td>
<td></td>
<td>description:Description</td>
</tr>
<tr>
<td>Productname</td>
<td></td>
<td>name:Name</td>
</tr>
<tr>
<td>ProductNumber</td>
<td></td>
<td>productnumber:ProductID</td>
</tr>
<tr>
<td>SalesUnitSymbol</td>
<td>Fn</td>
<td>defaultuomid:name:Default Unit (Name)</td>
</tr>
<tr>
<td>SalesPrice</td>
<td></td>
<td>price:UnitPrice</td>
</tr>
<tr>
<td>UnitCost</td>
<td></td>
<td>currentcost:Current Cost</td>
</tr>
<tr>
<td>IsStockedProduct</td>
<td>Fn</td>
<td>isstockitem:Stock Item</td>
</tr>
<tr>
<td>ProductType</td>
<td>Fn</td>
<td>producttypecode:Product Type</td>
</tr>
<tr>
<td>None</td>
<td></td>
<td>defaultuomscheduled:name:Unit Group (Name)</td>
</tr>
<tr>
<td>None</td>
<td></td>
<td>msdyn_ismaintainedexternally:Is Maintained Externally</td>
</tr>
<tr>
<td>SalesUnitDecimalPrecision</td>
<td>Fn</td>
<td>quantitydecimal:Decimals Supported</td>
</tr>
<tr>
<td>FieldServiceProductType</td>
<td>Fn</td>
<td>msdyn_fieldserviceproducttype:Field Service Product Type</td>
</tr>
</tbody>
</table>
This topic discusses the templates and underlying tasks that are used to synchronize work orders in Dynamics 365 Field Service to sales orders in Dynamics 365 Supply Chain Management.

Templates and tasks

The following templates and underlying tasks are used to run the synchronization of work orders in Field Service to sales orders in Supply Chain Management.

**Names of the templates in Data integration**

The **Work orders to Sales orders (Field Service to Supply Chain Management)** template is used to run synchronization.

**Names of the tasks in the Data integration project**

- WorkOrderHeader
- WorkOrderServiceLineEstimate
- WorkOrderServiceLineUsed
- WorkOrderProductLineEstimate
- WorkOrderProductLineUsed

The following synchronization tasks are required before synchronization of sales order headers and lines can occur:

- Field Service Products (Supply Chain Management to Field Service)
- Accounts (Sales to Supply Chain Management) – Direct

Entity set
## Entity flow

Work orders are created in Field Service. If the work orders include only externally maintained products, and if the Work order status value differs from Open-Unscheduled and Closed – Cancelled, the work orders can be synchronized to Supply Chain Management via a Microsoft Dataverse Data integration project. Updates on the work orders will be synchronized as sales orders in Supply Chain Management. These updates include the information about the origin type and status.

## Estimated versus Used

In Field Service, products and services on work orders have both Estimated values and Used values for quantities and amounts. However, in Supply Chain Management, sales orders don't have the same concept of Estimated and Used values. To support product allocation that uses the expected quantity on the sales order in Supply Chain Management, but to keep the used quantity that should be consumed and invoiced, two sets of tasks synchronize the products and services on the work order. One set of tasks is for Estimated values, and the other set of tasks is for Used values.

This behavior enables scenarios where estimated values are used for allocation or reservation in Supply Chain Management, whereas used values are used for consumption and invoicing.

### Estimated

For synchronization of product lines, the Estimated values are used when the Line Status value is Estimated, the Allocated option is set to Yes, and System Status value isn't Closed – Posted.

For synchronization of service lines, the Estimated values are used when the Line Status value is Estimated and the System Status value isn't Closed – Posted.

### Used

The Used values are used for consumption and invoicing. In these cases, the Used values are synchronized.

The following table provides an overview of the various combinations for product lines.

<table>
<thead>
<tr>
<th>SYSTEM STATUS (FIELD SERVICE)</th>
<th>LINE STATUS (FIELD SERVICE)</th>
<th>ALLOCATED (FIELD SERVICE)</th>
<th>SYNCHRONIZED VALUE (SUPPLY CHAIN MANAGEMENT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open - Scheduled</td>
<td>Estimated</td>
<td>Yes</td>
<td>Estimated</td>
</tr>
<tr>
<td>Open - Scheduled</td>
<td>Estimated</td>
<td>No</td>
<td>Used</td>
</tr>
<tr>
<td>Open - Scheduled</td>
<td>Used</td>
<td>Yes</td>
<td>Used</td>
</tr>
<tr>
<td>Open - Scheduled</td>
<td>Used</td>
<td>No</td>
<td>Used</td>
</tr>
<tr>
<td>Open - In Progress</td>
<td>Estimated</td>
<td>Yes</td>
<td>Estimated</td>
</tr>
</tbody>
</table>
The following table provides an overview of the various combinations for service lines:

<table>
<thead>
<tr>
<th>SYSTEM STATUS (FIELD SERVICE)</th>
<th>LINE STATUS (FIELD SERVICE)</th>
<th>ALLOCATED (FIELD SERVICE)</th>
<th>SYNCHRONIZED VALUE (SUPPLY CHAIN MANAGEMENT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open - In Progress</td>
<td>Estimated</td>
<td>No</td>
<td>Used</td>
</tr>
<tr>
<td>Open - In Progress</td>
<td>Used</td>
<td>Yes</td>
<td>Used</td>
</tr>
<tr>
<td>Open - In Progress</td>
<td>Used</td>
<td>No</td>
<td>Used</td>
</tr>
<tr>
<td>Open - Completed</td>
<td>Estimated</td>
<td>Yes</td>
<td>Estimated</td>
</tr>
<tr>
<td>Open - Completed</td>
<td>Estimated</td>
<td>No</td>
<td>Used</td>
</tr>
<tr>
<td>Open - Completed</td>
<td>Used</td>
<td>Yes</td>
<td>Used</td>
</tr>
<tr>
<td>Open - Completed</td>
<td>Used</td>
<td>No</td>
<td>Used</td>
</tr>
<tr>
<td>Closed - Posted</td>
<td>Estimated</td>
<td>Yes</td>
<td>Used</td>
</tr>
<tr>
<td>Closed - Posted</td>
<td>Estimated</td>
<td>No</td>
<td>Used</td>
</tr>
<tr>
<td>Closed - Posted</td>
<td>Used</td>
<td>Yes</td>
<td>Used</td>
</tr>
<tr>
<td>Closed - Posted</td>
<td>Used</td>
<td>No</td>
<td>Used</td>
</tr>
</tbody>
</table>

Synchronization of **Estimated** values versus **Used** values is managed through the two sets of tasks for product lines and service lines. Predefined filters trigger the correct task, and the underlying mapping helps guarantee that the correct values for **Expected** versus **Used** are synchronized.

**Example**

1. A work order is created and scheduled in Field Service.
The **System Status** value is **Open – Scheduled**.

- **Product line**: Estimated Qty = 5ea, Used Qty = 0ea, Line Status = Estimated, Allocated = No
- **Service line**: Estimated Qty = 2h, Used Qty = 0h, Line Status = Estimated

In this example, the product's **Used Qty** value of 0 (zero) and the service's **Estimated Qty** value of 2h are synchronized to Supply Chain Management.

2. Products are allocated in Field Service.

The **System Status** value is **Open – Scheduled**.

- **Product line**: Estimated Qty = 5ea, Used Qty = 0ea, Line Status = Estimated, Allocated = Yes
- **Service line**: Estimated Qty = 2h, Used Qty = 0h, Line Status = Estimated

In this example, the product's **Estimated Qty** value of 5ea and the service's **Estimated Qty** value of 2h are synchronized to Supply Chain Management.

3. The service technician starts to work on the work order and registers material usage of 6.

The **System Status** value is **Open – In Progress**.

- **Product line**: Estimated Qty = 5ea, Used Qty = 6ea, Line Status = Used, Allocated = Yes
- **Service line**: Estimated Qty = 2h, Used Qty = 0h, Line Status = Estimated

In this example, the product's **Used Qty** value of 6 and the service's **Estimated Qty** value of 2h are synchronized to Supply Chain Management.

4. The service technician completes the work order and registers used time of 1.5 hours.

The **System Status** value is **Open – Completed**.

- **Product line**: Estimated Qty = 5ea, Used Qty = 6ea, Line Status = Used, Allocated = Yes
- **Service line**: Estimated Qty = 2h, Used Qty = 1.5h, Line Status = Used

In this example, the product's **Used Qty** value of 6 and the service's **Used Qty** of 1.5h are synchronized to Supply Chain Management.

**Sales order origin and status**

**Sales origin**

To keep track of sales orders that originate from work orders, you can create a sales origin where the **Origin type assignment** option is set to **Yes** and the **Sales origin type** field is set to **Work order integration**.

By default, the mapping selects the sales origin for the **Work order integration** sales origin type for all sales orders that are created from work orders. This behavior can be useful when you work with the sales order in Supply Chain Management. You must make sure that sales orders that originate from work orders aren't synchronized back to Field Service as work orders.

For details about how to create the correct sales origin setup in Supply Chain Management, see the "Preconditions and mapping setup" section of this topic.

**Status**

When the sales order originates from a work order, the **External work order status** field appears on the **Setup** tab on the sales order header. This field shows the system status from the work order in Field Service, to help track the synchronized work order status of sales orders in the Supply Chain Management. This field can also help the user determine when the sales order should be shipped or invoiced.

The **External work order status** field can have the following values:

- Open - Scheduled
Field Service CRM solution

To support the integration between Field Service and Supply Chain Management, additional functionality from the Field Service CRM solution is required. The solution includes the following changes.

Work Order entity

The Has Externally Maintained Products Only field has been added to the Work Order entity and appears on the page. It's used to consistently track whether a work order consists entirely of externally maintained products. A work order consists entirely of externally maintained products when all the related products are maintained in Supply Chain Management. This field helps guarantee that users don't synchronize work orders that have products that are unknown.

Work Order Product entity

- The Order Has Externally Maintained Products Only field has been added to the Work Order Product entity and appears on the page. It's used to consistently track whether the work order product is maintained in Supply Chain Management. This field helps guarantee that users don't synchronize work order products that are unknown to Supply Chain Management.
- The Header System Status field has been added to the Work Order Product entity and appears on the page. It's used to consistently track the system status of the work order and helps guarantee correct filtering when work order products are synchronized to Supply Chain Management. When filters are set on the integration tasks, Header System Status information is also used to determine whether the estimated or used values should be synchronized.
- The Invoiced Unit Amount field shows the amount that is invoiced per actual unit that is used. The value is calculated as the Total Amount value divided by the Actual Quantity value. The field is used for integration to systems that don't support different values for the used quantity and the billed quantity. This field doesn't appear in the user interface (UI).
- The Invoiced Discount Amount field is calculated as the Discount Amount value plus the rounding from the calculation of the Invoiced Unit Amount value. This field is used for integration and doesn't appear in the UI.
- The Decimal Quantity field stores the value from the Quantity field as a decimal number. This field is used for integration and doesn't appear in the UI.
- The value in Used fields is reset to 0 (zero) if the Line Status value of the work order product is changed from Used to Estimated. This change helps prevent situations where a used quantity that is mistakenly entered is used for synchronization when the Line Status value is Estimated and the Allocated option is set to No.

Work Order Service entity

- The Order Has Externally Maintained Products Only field has been added to the Work Order Service entity and appears on the page. It's used to consistently track whether the work order service is maintained in Supply Chain Management. This field helps guarantee that users don't synchronize work order services that are unknown to Supply Chain Management.
- The Header System Status field has been added to the Work Order Service entity and appears on the page. It's used to consistently track the system status of the work order and helps guarantee correct filtering when work order services are synchronized to Supply Chain Management. When filters are set on the integration tasks, Header System Status information is also used to determine whether the estimated or used values should be synchronized.
- The Duration In Hours field stores the value from the Duration field after that value is converted from minutes to hours. This field is used for integration and doesn't appear in the UI.
- The **Estimated Duration In Hours** field stores the value from the **Estimated Duration** field after that value is converted from minutes to hours. This field is used for integration and doesn't appear in the UI.

- The **Invoiced Unit Amount** field stores the amount that is invoiced per actual unit that is used. The value is calculated as the **Total Amount** value divided by the **Actual Quantity** value. This field is used for integration to systems that don't support different values for the used quantity and the billed quantity. The field doesn't appear in the UI.

- The **Invoiced Discount Amount** field is calculated as the **Discount Amount** value plus the rounding from the calculation of the **Invoiced Unit Amount** value. This field is used for integration and doesn't appear in the UI.

- The **External Line Order** field is a calculated negative line order number that can be used in external systems where work order product and service lines are combined. This field enables work order products that are inserted to have positive line numbers and work order services to have negative line numbers. The value of this field is calculated as the **Line Order** value multiplied by -1. This field doesn't appear in the UI.

- The value in **Used** fields is reset to 0 (zero) if the **Line Status** value of the work order service is changed from **Used** to **Estimated** for some reason. This change helps prevent situations where a used quantity that is mistakenly entered is used for synchronization when the **Line Status** value is **Estimated** and the **Header System Status** value is **Closed – Posted**.

### Preconditions and mapping setup

Before you synchronize work orders, it's important that you update the following settings in the systems.

**Setup in Field Service**

- Make sure that the number series that is used for work orders in Field Service doesn't overlap the number sequence that is used for sales orders in Supply Chain Management. Otherwise, existing sales orders can be incorrectly updated in Field Service or Supply Chain Management.

- The **Work Order Invoice Creation** field must be set to **Never**, because the invoicing will be done from Supply Chain Management. Go to **Field Service > Settings > Administration > Field Service Settings**, and make sure that the **Work Order Invoice Creation** field is set to **Never**.

**Setup in Supply Chain Management**

Work order integration requires that you set up the sales origin. The sales origin is used to distinguish sales orders in Supply Chain Management that were created from work orders in Field Service. When a sales order has a sales origin of the **Work order integration** type, the **External work order status** field appears on the sales order header. Additionally, the sales origin helps guarantee that sales orders that were created from work orders in Field Service are filtered out during sales order synchronization from Supply Chain Management to Field Service.

1. Go to **Sales and marketing > Setup > Sales orders > Sales origin**.
2. Select **New** to create a new sales origin.
3. In the **Sales origin** field, enter a name for the sales origin, such as **WorkOrder**.
4. In the **Description** field, enter a description, such as **Field Service Work Order**.
5. Select the **Origin type assignment** check box.
6. Set the **Sales origin type** field to **Work order integration**.
7. Select **Save**.

**Setup in Data integration**

Ensure the **Integration key** exist for **msdyn_workorders**

1. Go to Data Integration
2. Select **Connection Set** tab
3. Select the Connection set used for Work order synchronization
4. Select Integration key tab
5. Find msdyn_workorders and check that the key msdyn_name (Work Order Number) is added. If it is not shown, add it by click Add key and click Save in the top of the page

Template mapping in Data integration
The following illustrations show the template mapping in Data integration.

Work orders to Sales orders (Field Service to Supply Chain Management): WorkOrderHeader
Filter: (msdyn_systemstatus ne 690970005) and (msdyn_systemstatus ne 690970000) and (msdynce_hasexternallymaintainedproductsonly eq true)

<table>
<thead>
<tr>
<th>Source Field</th>
<th>Map Type</th>
<th>Destination Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>msdyn_serviceaccount.accountnumber</td>
<td>=</td>
<td>OrderingCustomerAccountNumber</td>
</tr>
<tr>
<td>msdyn_city</td>
<td>=</td>
<td>DeliveryAddressCity</td>
</tr>
<tr>
<td>msdyn_country</td>
<td>=</td>
<td>DeliveryAddressCountry</td>
</tr>
<tr>
<td>transactioncurrencyid</td>
<td>=</td>
<td>CurrencyCode</td>
</tr>
<tr>
<td>msdyn_address1</td>
<td>=</td>
<td>DeliveryAddressStreet</td>
</tr>
<tr>
<td>msdyn_name</td>
<td>=</td>
<td>SalesOrderNumber</td>
</tr>
<tr>
<td>msdyn_postalcode</td>
<td>=</td>
<td>DeliveryAddressZipCode</td>
</tr>
<tr>
<td>msdyn_address2</td>
<td>=</td>
<td>DeliveryAddressStreetNumber</td>
</tr>
<tr>
<td>msdyn_stateprovince</td>
<td>=</td>
<td>DeliveryAddressStateId</td>
</tr>
<tr>
<td>msdyn_billingaccount.accountnumber</td>
<td>=</td>
<td>InvoiceCustomerAccountNumber</td>
</tr>
<tr>
<td>None</td>
<td>=</td>
<td>DeliveryAddressName</td>
</tr>
<tr>
<td>None</td>
<td>=</td>
<td>DeliveryAddressOrderSpecific</td>
</tr>
<tr>
<td>None</td>
<td>=</td>
<td>DeliveryAddressDescription</td>
</tr>
<tr>
<td>None</td>
<td>=</td>
<td>SalesOrderOriginType</td>
</tr>
<tr>
<td>msdyn_systemstatus</td>
<td>=</td>
<td>ExternalWorkOrderStatus</td>
</tr>
</tbody>
</table>

Work orders to Sales orders (Field Service to Supply Chain Management): WorkOrderServiceLineEstimate
Filter: (msdynce_headersystemstatus ne 690970005) and (msdynce_headersystemstatus ne 690970000) and (msdynce_orderhasexternallymaintainedproductsonly eq true) and (msdyn_linestatus eq 690970000) and (msdynce_headersystemstatus ne 690970004)
### Work orders to Sales orders (Field Service to Supply Chain Management): WorkOrderServiceLineUsed

Filter: (msdynce_headersystemstatus ne 690970005) and (msdynce_headersystemstatus ne 690970000) and (msdynce_orderhasexternalmaintainedproductsonly eq true) and ((msdyn_linestatus eq 690970001) or (msdynce_headersystemstatus eq 690970004))

### Work orders to Sales orders (Field Service to Supply Chain Management): WorkOrderProductLineEstimate

Filter: (msdynce_headersystemstatus ne 690970005) and (msdynce_headersystemstatus ne 690970000) and (msdynce_orderhasexternalmaintainedproductsonly eq true) and (msdyn_linestatus eq 690970000) and (msdynce_headersystemstatus ne 690970004) and (msdyn_allocated eq true)
Work orders to Sales orders (Field Service to Supply Chain Management): WorkOrderProductLineUsed
Filter: (msdynce_headersystemstatus ne 690970005) and (msdynce_headersystemstatus ne 690970000) and (msdynce_orderhasexternalmaintainedproductsonly eq true) and ((msdyn_linestatus eq 690970001) or (msdynce_headersystemstatus eq 690970004) or (msdyn_allocated ne true))

<table>
<thead>
<tr>
<th>Source Field</th>
<th>Map Type</th>
<th>Destination Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>msdyn_product.productnumber</td>
<td>=</td>
<td>ProductNumber [ProductNumber]</td>
</tr>
<tr>
<td>msdyn_estimatedquantity</td>
<td>=</td>
<td>OrderedSalesQuantity [OrderedSalesQuantity]</td>
</tr>
<tr>
<td>msdyn_workorder.msdyn_name</td>
<td>=</td>
<td>SalesOrderNumber [SalesOrderNumber]</td>
</tr>
<tr>
<td>msdyn_lineorder</td>
<td>=</td>
<td>LineCreationSequenceNumber [LineCreationSequenceNumber]</td>
</tr>
<tr>
<td>msdyn_unitnumber</td>
<td>fn</td>
<td>SalesUnitSymbol [SalesUnitSymbol]</td>
</tr>
<tr>
<td>msdyn_estimatedunitamount</td>
<td>=</td>
<td>SalesPrice [SalesPrice]</td>
</tr>
<tr>
<td>msdyn_estimateddiscountamount</td>
<td>=</td>
<td>TotalDiscountAmount [TotalDiscountAmount]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source Field</th>
<th>Map Type</th>
<th>Destination Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>msdyn_product.productnumber</td>
<td>=</td>
<td>ProductNumber [ProductNumber]</td>
</tr>
<tr>
<td>msdyn_quantity</td>
<td>=</td>
<td>OrderedSalesQuantity [OrderedSalesQuantity]</td>
</tr>
<tr>
<td>msdyn_workorder.msdyn_name</td>
<td>=</td>
<td>SalesOrderNumber [SalesOrderNumber]</td>
</tr>
<tr>
<td>msdyn_lineorder</td>
<td>=</td>
<td>LineCreationSequenceNumber [LineCreationSequenceNumber]</td>
</tr>
<tr>
<td>msdyn_unitname</td>
<td>fn</td>
<td>SalesUnitSymbol [SalesUnitSymbol]</td>
</tr>
<tr>
<td>msdyn_invoicedunitamount</td>
<td>=</td>
<td>SalesPrice [SalesPrice]</td>
</tr>
<tr>
<td>msdyn_invoiceddiscountamount</td>
<td>=</td>
<td>TotalDiscountAmount [TotalDiscountAmount]</td>
</tr>
</tbody>
</table>
This topic discusses the templates and underlying tasks that are used to synchronize agreement invoices in Dynamics 365 Field Service to free text invoices in Dynamics 365 Supply Chain Management.

**Templates and tasks**

The following template and underlying tasks are used to run the synchronization of agreement invoices from Field Service to free text invoices in Supply Chain Management.

**Name of the template in Data integration**

- Agreement invoices (Field Service to Supply Chain Management)

**Names of the tasks in the Data integration project**

- Invoice headers
- Invoice lines

The following synchronization is required before the synchronization of agreement invoices can occur:

- Accounts (Sales to Supply Chain Management) – Direct

**Entity set**

<table>
<thead>
<tr>
<th>FIELD SERVICE</th>
<th>SUPPLY CHAIN MANAGEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>invoices</td>
<td>Dataverse customer free text invoice headers</td>
</tr>
<tr>
<td>invoicedetails</td>
<td>Dataverse customer free text invoice lines</td>
</tr>
</tbody>
</table>

**Entity flow**

Invoices that are created from an agreement in Field Service can be synchronized to Supply Chain Management via a Microsoft Dataverse Data integration project. Updates to these invoices will be synchronized to the free text invoices in Supply Chain Management if the accounting status of the free text invoices is **In process**. After the free text invoices are posted in Supply Chain Management, and the accounting status is updated to **Completed**, you cannot synchronize updates from Field Service.

**Field Service CRM solution**

The **Has Lines With Agreement Origin** column has been added to the **Invoice** table. This column helps guarantee that only invoices that are created from an agreement are synchronized. The value is **true** if the invoice contains at least one invoice line that originates from an agreement.

The **Has Agreement Origin** column has been added to the **Invoice Line** table. This column helps guarantee that only invoice lines that are created from an agreement are synchronized. The value is **true** if the invoice line originates from an agreement.
**Invoice date** is a mandatory field in Supply Chain Management. Therefore, the column must have a value in Field Service before synchronization occurs. To meet this requirement, the following logic is added:

- If the **Invoice Date** column is blank on the **Invoice** table (that is, if it has no value), it's set to the current date when an invoice line that originates from an agreement is added.
- The user can change the **Invoice Date** column. However, when the user tries to save an invoice that originates from an agreement, they receive a business process error if the **Invoice Date** column is blank on the invoice.

**Prerequisites and mapping setup**

**In Supply Chain Management**

An invoice origin must be set up for the integration, to distinguish the free text invoices in Supply Chain Management that are created from agreement invoices in Field Service. When an invoice has an invoice origin of the **Agreement invoice integration** type, the **External invoice number** field is shown on the **Sales invoice** header.

Besides appearing on the invoice header, the **External invoice number** information can be used to help guarantee that invoices that are created from Field Service agreement invoices are filtered out during invoice synchronization from Supply Chain Management to Field Service.

1. Go to **Accounts receivable** > **Setup** > **Invoice origin codes**.
2. Select **New** to create a new invoice origin.
3. In the **Invoice origin** field, enter a name for the invoice origin, such as **FS**.
4. In the **Description** field, enter a description, such as **Field Service Agreement Invoice**.
5. Select the **Origin type assignment** check box.
6. Set the **Invoice origin type** field to **Agreement invoice integration**.
7. Select **Save**.

**In the Data Integration project**

Task: **Invoice lines**

Make sure that the default value for the Supply Chain Management field **Main Account Display Value** is updated to match the desired value.

The default template value is **401100**.

**Template mapping in Data integration**

The following illustrations show the template mapping in Data integration.

**Agreement invoices (Field Service to Supply Chain Management): Invoice headers**
### Agreement invoices (Field Service to Supply Chain Management): Invoice lines

<table>
<thead>
<tr>
<th>Source Field</th>
<th>Map Type</th>
<th>Destination Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>duedate [Due Date]</td>
<td>=</td>
<td>DueDate [DueDate]</td>
</tr>
<tr>
<td>invoicenumber [Invoice ID]</td>
<td>=</td>
<td>ExternallInvoiceld [ExternallInvoiceld]</td>
</tr>
<tr>
<td>customerId:Account[accountnumber]:Contact[msdyn_contactnumber]..</td>
<td>=</td>
<td>CustomerAccount [CustomerAccount]</td>
</tr>
<tr>
<td>msdyn_invoiceDate [Invoice Date]</td>
<td>=</td>
<td>InvoiceDate [InvoiceDate]</td>
</tr>
<tr>
<td>customerId:Account[accountnumber]:Contact[msdyn_contactnumber]..</td>
<td>=</td>
<td>InvoiceAccount [InvoiceAccount]</td>
</tr>
</tbody>
</table>

None  

<table>
<thead>
<tr>
<th>Source Field</th>
<th>Map Type</th>
<th>Destination Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>productdescription [Write-In Product]</td>
<td>=</td>
<td>Description [Description]</td>
</tr>
<tr>
<td>quantity [Quantity]</td>
<td>=</td>
<td>Quantity [Quantity]</td>
</tr>
<tr>
<td>sequencenumber [Sequence Number]</td>
<td>=</td>
<td>LineNumber [LineNumber]</td>
</tr>
<tr>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>invoiced invoicenumber [Invoice ID [Invoice ID]]</td>
<td>=</td>
<td>ExternallInvoiceld [ExternallInvoiceld]</td>
</tr>
<tr>
<td>priceperunit [Price Per Unit]</td>
<td>=</td>
<td>UnitPrice [UnitPrice]</td>
</tr>
</tbody>
</table>
This topic discusses the templates and underlying tasks that are used to synchronize inventory adjustments and transfers from Dynamics 365 Supply Chain Management to Dynamics 365 Field Service.

### Templates and tasks

The following template and underlying tasks are used to synchronize inventory adjustments and transfers from Field Service to Supply Chain Management.

**Templates in Data integration**

- Inventory Adjustment (Field Service to Supply Chain Management)
- Inventory Transfers (Field Service to Supply Chain Management)

**Tasks in the Data integration projects**

- Inventory Adjustments
- Inventory Transfers

### Table set

<table>
<thead>
<tr>
<th>FIELD SERVICE</th>
<th>SUPPLY CHAIN MANAGEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>msdyn_inventoryadjustmentproducts</td>
<td>Dataverse Inventory adjustment journal headers and lines</td>
</tr>
<tr>
<td>msdyn_inventoryadjustmentproducts</td>
<td>Dataverse inventory transfer journal headers and lines</td>
</tr>
</tbody>
</table>

### Table flow

Inventory adjustments and transfers made in Field Service will synchronize to Supply Chain Management after the **Post status** changes from **Created** to **Posted**. When this occurs, the adjustment or the transfer order will be locked and become read only. This means that adjustments and transfers can be posted in Supply Chain
Field Service CRM solution

The **Inventory unit** column has been added to the **Product** table. This column is needed because the Sales and Inventory unit is not always the same in Supply Chain Management, and the Inventory Unit is needed for the Warehouse Inventory in Supply Chain Management. When you set the product on an Inventory adjustment product for both Inventory adjustments and Inventory transfers, the unit will be fetched from the inventory product value. If a value is found, the **Unit** column will be locked on the Inventory adjustment product.

The **Post status** column has been added to both the **Inventory adjustment** table and the **Inventory transfer** table. This column is used as a filter when an adjustment or transfer is sent to Supply Chain Management. The default for this column is Created (1), however it is not sent to Supply Chain Management. When you update the value to Posted (2), it is sent to Supply Chain Management, but after that you will no longer be able to change the adjustment or transfer or add new lines.

The **Number sequence** column has been added to the **Inventory adjustment product** table. This column ensures that the integration has a unique number, so the integration can create and update the adjustment. When you create your first inventory adjustment product, it will create a new record in the **P2C AutoNumber** table to maintain the number series and the prefix that is used.

Prerequisites and mapping setup

**Supply Chain Management**

The integration inventory journals generated by the integration can automatically be posted using a batch job. This is enabled from **Inventory management > Periodic tasks > Dataverse integration > Post integration inventory journals**.

**Template mapping in Data integration**

The following illustrations show the template mapping in Data integration.

**Inventory adjustment (Field Service to Supply Chain Management): Inventory adjustment**

![Inventory adjustment mapping](image)

**Inventory transfer (Field Service to Supply Chain Management): Inventory transfer**

![Inventory transfer mapping](image)
<table>
<thead>
<tr>
<th>Source Field</th>
<th>Map Type</th>
<th>Destination Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>msdyn_inventoryadjustmentproducts</td>
<td></td>
<td>LINENUMBER [LINENUMBER]</td>
</tr>
<tr>
<td>msdyn_quantity [msdyn_quantity]</td>
<td></td>
<td>INVENTORYQUANTITY [INVENTORYQUANTITY]</td>
</tr>
<tr>
<td>msdyn_inventorytransfer.msdyn_name [msdyn_inventorytransfer._name]</td>
<td></td>
<td>JOURNALNUMBER [JOURNALNUMBER]</td>
</tr>
<tr>
<td>msdyn_product.productnumber [msdyn_productproductnumber]</td>
<td></td>
<td>PRODUCTNUMBER [PRODUCTNUMBER]</td>
</tr>
<tr>
<td>DestinationWarehouseName [DestinationWarehouseName]</td>
<td></td>
<td>DESTINATIONWAREHOUSEID [DESTINATIONWAREHOUSEID]</td>
</tr>
<tr>
<td>SourceWarehouseName [SourceWarehouseName]</td>
<td></td>
<td>SOURCEWAREHOUSEID [SOURCEWAREHOUSEID]</td>
</tr>
<tr>
<td>TransferOrderJournalType [TransferOrderJournalType]</td>
<td></td>
<td>JOURNALNAMED [JOURNALNAMED]</td>
</tr>
</tbody>
</table>
This topic discusses the templates and underlying tasks that are used to synchronize inventory-level information from Dynamics 365 Supply Chain Management to Dynamics 365 Field Service.

Templates and tasks

The following template and underlying tasks are used to synchronize inventory on-hand levels from Supply Chain Management to Field Service.

**Template in Data integration**

- Product Inventory (Supply Chain Management to Field Service)

**Task in the Data integration project**

- Product inventory

The following synchronization tasks are required before synchronization of inventory levels can occur:

- Warehouses (Supply Chain Management to Field Service)
- Field Service products with Inventory unit (Supply Chain Management to Sales)

**Entity set**

<table>
<thead>
<tr>
<th>FIELD SERVICE</th>
<th>SUPPLY CHAIN MANAGEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>msdynce_externalproductinventories</td>
<td>Dataverse inventory on-hand by warehouse</td>
</tr>
</tbody>
</table>

**Entity flow**

Inventory-level information from Finance and Operation is sent to Field Service for selected products. The inventory-level information includes:
- On hand quantity (current recorded physical quantity located in the warehouse)
- On order quantity (total recorded quantity on order, such as sales orders)
- Ordered quantity (total recorded ordered quantity, such as purchase orders)

This information is captured per released product for each warehouse and synchronized based on change tracking, when the inventory level changes.

In Field Service, the integration solution creates inventory journals for the delta, so that the levels in Field Service match the levels in Supply Chain Management.

Supply Chain Management will act as the master for inventory levels. Therefore it is important to set up integration for work orders, transfers, and adjustments from Field Service to Supply Chain Management if this functionality is used in Field Service, together with synchronization of inventory levels from Supply Chain Management.

The products and warehouses where inventory levels are mastered from Supply Chain Management can be controlled with the Advanced Query and Filtering (Power Query).

NOTE
It is possible to create multiple warehouses in Field Services (with Is Externally Maintained = No) and then map them to a single warehouse in Supply Chain Management, with the Advanced query and filtering functionality. This is used in situations where you want Field Service to master the detailed inventory level and only send updates to Supply Chain Management. In this case, Field Service will not receive inventory-level updates from Supply Chain Management. For additional information, see Synchronize inventory adjustments from Field Service to Supply Chain Management and Synchronize work orders in Field Service to sales orders linked to project in Supply Chain Management.

Field Service CRM solution

The External product inventory entity is only used for back end in to the integration. This entity receives the inventory level values from Supply Chain Management in the integration and then transforms those values to Manual inventory journals, which then changes the Inventory products in the Warehouse.

Prerequisites and mapping setup

**Data integration**

For the project to work, you need to ensure that the Integration key is updated for msdynce_externalproductinventories.

1. Go to Data integration > Connection sets.
2. Select the used Connection set.
3. On the Integration key tab, ensure that the following keys are added to msdynce_externalproductinventories:
   - msdynce_productnumber (Product Number)
   - msdynce_warehouseid (Warehouse ID)

**Data integration project**

You can apply filters with Advanced Query and Filtering so that only certain products and warehouses send inventory-level information from Supply Chain Management to Field Service.

Template mapping in Data integration

**Product inventory (Supply Chain Management to Field Service): Product inventory**
<table>
<thead>
<tr>
<th>Source Field</th>
<th>Map Type</th>
<th>Destination Field</th>
<th>Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVENTORYWAREHOUSEID</td>
<td></td>
<td>msdynce_warehouse.msdyn_name (Warehouse</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>N)</td>
<td></td>
</tr>
<tr>
<td>PRODUCTNUMBER</td>
<td></td>
<td>msdynce_productnumber [Product Number]</td>
<td></td>
</tr>
<tr>
<td>ONHANDQUANTITY</td>
<td></td>
<td>msdynce_onhandquantity [On-hand Quantity]</td>
<td></td>
</tr>
<tr>
<td>ONORDERQUANTITY</td>
<td></td>
<td>msdynce_onorderquantity [On Order Quantity]</td>
<td></td>
</tr>
<tr>
<td>ORDEREDQUANTITY</td>
<td></td>
<td>msdynce_orderedquantity [Ordered Quantity]</td>
<td></td>
</tr>
<tr>
<td>PRODUCTNUMBER</td>
<td></td>
<td>msdynce_product.productnumber [Product</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Product Number)]</td>
<td></td>
</tr>
<tr>
<td>INVENTORYWAREHOUSEID</td>
<td></td>
<td>msdynce_warehouseid [Warehouse ID]</td>
<td></td>
</tr>
</tbody>
</table>
This topic discusses the templates and underlying task that are used to synchronize products with inventory unit from Dynamics 365 Supply Chain Management to Dynamics 365 Field Service.

The used Field Service Products with Inventory unit (Supply Chain Management to Field Service) template is based on the Field Service Products (Supply Chain Management to Field Service) template. For more information, see Synchronize products in Supply Chain Management to products in Field Service.

This topic only describes the differences between the two templates:

- Field Service Products with Inventory unit (Supply Chain Management to Sales)
- Field Service Products (Supply Chain Management to Field Service)

Templates and tasks

Name of the template in Data integration:
- Field Service Products with Inventory unit (Supply Chain Management to Sales)

Name of the task in the Data integration project:
- Products

The Field Service Products with Inventory unit (Supply Chain Management to Field Service) template includes one mapping that isn't included in the Field Service Products (Supply Chain Management to Field Service) template. This mapping ensures that the Inventory unit needed for inventory level synchronization is included.

```
INVENTORYUNITSYMBOL [INVENTORYUNITSYMBOL] Fn msdynce_inventoryunit.name [Inventory Unit(Name)]
```

Template mapping in Data integration
The following illustrations show the template mapping in Data integration.

### Field Service Products with Inventory unit (Supply Chain Management to Field Service): Products

<table>
<thead>
<tr>
<th>Source Field</th>
<th>Map Type</th>
<th>Destination Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURRENCYCODE [CURRENCYCODE]</td>
<td>=</td>
<td>transactionscurrencyid.iscurrencycode [Currency Code]</td>
</tr>
<tr>
<td>PRODUCTDESCRIPTION [PRODUCTDESCRIPTION]</td>
<td>=</td>
<td>description [Description]</td>
</tr>
<tr>
<td>PRODUCTNAME [PRODUCTNAME]</td>
<td>=</td>
<td>name [Name]</td>
</tr>
<tr>
<td>PRODUCTNUMBER [PRODUCTNUMBER]</td>
<td>=</td>
<td>productnumber [Product ID]</td>
</tr>
<tr>
<td>SALESUNITSYMBOL [SALESUNITSYMBOL]</td>
<td>Fn</td>
<td>defaultucmid.name [Default Unit (Name)]</td>
</tr>
<tr>
<td>SALESPRICE [SALESPRICE]</td>
<td>=</td>
<td>price [List Price]</td>
</tr>
<tr>
<td>UNITCOST [UNITCOST]</td>
<td>=</td>
<td>unitcost [Current Cost]</td>
</tr>
<tr>
<td>ISTOPPEDPRODUCT [ISTOPPEDPRODUCT]</td>
<td>Fn</td>
<td>isstockeditem [Stock Item]</td>
</tr>
<tr>
<td>PRODUCTTYPE [PRODUCTTYPE]</td>
<td>Fn</td>
<td>producttypecode [Product Type]</td>
</tr>
<tr>
<td>None</td>
<td>Fn</td>
<td>defaultomscheduled.name [Unit Group (Name)]</td>
</tr>
<tr>
<td>None</td>
<td>Fn</td>
<td>msdynce.ismaintainedexternally [Is Maintained Externally]</td>
</tr>
<tr>
<td>SALESUNITDECIMALPRECISION [SALESUNITDECIMALPRECISION]</td>
<td>Fn</td>
<td>quantitydecimal [Decimals Supported]</td>
</tr>
<tr>
<td>FIELDSERVICEPRODUCTTYPE [FIELDSERVICEPRODUCTTYPE]</td>
<td>Fn</td>
<td>msdyn_fieldserviceproducttype [Field Service Product Type]</td>
</tr>
<tr>
<td>INVENTORYUNITSYMBOL [INVENTORYUNITSYMBOL]</td>
<td>Fn</td>
<td>msdynce_inventoryunit.name [Inventory Unit (Name)]</td>
</tr>
</tbody>
</table>
This topic discusses the templates and underlying tasks that are used to synchronize projects from Dynamics 365 Supply Chain Management to Dynamics 365 Field Service.

**Templates and tasks**

The following template and underlying tasks are used to run synchronization of projects from Supply Chain Management to Field Service.

**Template in Data integration**

- Projects (Supply Chain Management to Field Service)

**Task in the Data integration project**

- Projects

The following synchronization tasks are required before synchronization of project list can occur:

- Accounts (Sales to Supply Chain Management)

**Entity set**

<table>
<thead>
<tr>
<th>FIELD SERVICE</th>
<th>SUPPLY CHAIN MANAGEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>msdynce_externalprojects</td>
<td>Projects</td>
</tr>
</tbody>
</table>

**Entity flow**

Projects are created in Supply Chain Management. Projects with **Project type** set to **Time and material** and **Project stage** set to **In process** will synchronize to the **External Project** entity in Field Service, including Project number, Project name, Project stage, and Customer account information. The **External Project** list is used to pair Field service work orders with Supply Chain Management projects.
Field Service CRM solution

The **External Project** entity gets all the projects from Supply Chain Management. The **External Project** field has been added to the **Work Order** entity. This is a lookup field, so by tagging your work order with a project, the sales order will be connected to a project within Supply Chain Management. After the **System Status** changes **Open – In Progress(690,970,000)** to a higher status, the **External Project** field will be locked and you can no longer add, remove, or change the value.

Prerequisites and mapping setup

**Supply Chain Management**

Enable change tracking for Data entity projects.

Template mapping in Data integration

**Projects (Supply Chain Management to Field Service): Projects**

<table>
<thead>
<tr>
<th>Source Field</th>
<th>Map Type</th>
<th>Destination Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROJECTNAME (PROJECTNAME)</td>
<td></td>
<td>msdynce_projectname [Name]</td>
</tr>
<tr>
<td>PROJECTID (PROJECTID)</td>
<td></td>
<td>msdynce_projectnumber [Project Number]</td>
</tr>
<tr>
<td>PROJECTSTAGE (PROJECTSTAGE)</td>
<td>F</td>
<td>msdynce_projectstage [Project Stage]</td>
</tr>
<tr>
<td>CUSTOMERACCOUNT (CUSTOMERACCOUNT)</td>
<td></td>
<td>msdynce_account.accountnumber [Account (Account Number)]</td>
</tr>
</tbody>
</table>
This topic discusses the templates and underlying tasks that are used to synchronize warehouses from Dynamics 365 Supply Chain Management to Dynamics 365 Field Service.

**Templates and tasks**

The following template and underlying tasks are used to run synchronization of warehouses from Supply Chain Management to Field Service.

**Template in Data integration**

- **Warehouses (Supply Chain Management to Field Service)**

**Task in the Data integration project**

- **Warehouse**

**Table set**

<table>
<thead>
<tr>
<th>FIELD SERVICE</th>
<th>SUPPLY CHAIN MANAGEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>msdyn_warehouses</td>
<td>Warehouses</td>
</tr>
</tbody>
</table>

**Table flow**

Warehouses that are created and maintained in Supply Chain Management can be synchronized to Field Service via a Microsoft Dataverse Data integration project. The warehouses that you want to synchronize to Field Service can be controlled with the Advanced query and filtering on the project. Warehouses that synchronize from Supply Chain Management are created in Field Service with the **Is maintained externally** column set to **Yes** and the record is read only.

**Field Service CRM solution**
To support the integration between Field Service and Supply Chain Management, additional functionality from the Field Service CRM solution is required. In the solution, the **Is Maintained Externally** column has been added to the Warehouse (msdyn_warehouses) table. This column helps to identify if the warehouse is handled from Supply Chain Management or if it only exists in Field Service. The settings for this column include:

- **Yes** – The warehouse originated from Supply Chain Management and won’t be editable in Sales.
- **No** – The warehouse was entered directly in Field Service and is maintained here.

The **Is Externally Maintained** column helps control the synchronization of inventory levels, adjustments, transfers, and usage on work orders. Only warehouses with **Is Externally Maintained** set to **Yes** can be used to synchronize directly to the same warehouse in the other system.

**NOTE**

It is possible to create multiple warehouses in Field Service (with **Is Externally Maintained** = No) and then map them to a single warehouse, with the Advanced query and filtering functionality. This is used in situations where you want Field Service to master the detailed inventory level and just send updates to Supply Chain Management. In this case, Field Service will not receive inventory-level updates from Supply Chain Management. For additional information, see Synchronize inventory adjustments from Field Service to Finance and Operations and Synchronize work orders in Field Service to sales orders linked to project in Finance and Operations.

### Prerequisites and mapping setup

#### Data Integration project

Before synchronizing the warehouses, make sure to update the Advanced query and filtering on the project to only include the warehouses that you want to bring from Supply Chain Management to Field Service. Note that you will need the warehouse in Field Service to apply it on work orders, adjustments, and transfers.

To ensure that the Integration key exists for msdyn_warehouses:

1. Go to Data Integration.
2. Select the Connection Set tab.
3. Select the connection set used for work order synchronization.
4. Select the Integration key tab.
5. Find msdyn_warehouses and confirm that the key msdyn_name (name) is added. If it is not shown, add it by clicking Add key and then click Save at the top of the page.

#### Template mapping in Data integration

The following illustration shows the template mapping in Data integration.

**Warehouses (Supply Chain Management to Field Service): Warehouse**
This topic discusses the templates and underlying task that are used to synchronize work orders with a project number from Dynamics 365 Field Service to Dynamics 365 Supply Chain Management.

The used *Work Orders with Project (Field Service to Supply Chain Management)* template is based on the *Work Orders (Field Service to Supply Chain Management)* template. For more information, see *Synchronize work orders in Field Service to sales orders in Supply Chain Management*.

This topic only describes the differences between the two templates:

- **Work Orders with Project (Field Service to Supply Chain Management)**
- **Work Orders (Field Service to Supply Chain Management)**

The main difference is that this template includes mapping of the project number assigned to the Work order in Field Service, ensuring that the Sales order created in Supply Chain Management include the project number and that invoicing can happen on the related project. Besides this the template use Advanced Query and Filtering.

**Templates and tasks**

**Name of the template in Data integration:**
- Work Orders with Project (Field Service to Supply Chain Management)

**Name of the task in the Data integration project:**
- WorkOrderHeader
- WorkOrderHeaderProject
- WorkOrderProduct
- WorkOrderService

**Field Service CRM solution**
The **External Project** field has been added to the Work Order entity. This field is a lookup and buy tagging your Work Order with a project the Sales Order will then be connected to a Project within Supply Chain Management. When the **System Status** changes from Open – In Progress (690,970,000) to a higher status, the **External Project** field will be locked and you can't add, remove, or change the value.

**Template mapping in Data integration**

The following illustrations show the template mapping in Data integration.

**Work Orders with Project (Field Service to Supply Chain Management): WorkOrderHeader**

<table>
<thead>
<tr>
<th>SOURCE FIELD</th>
<th>MAP TYPE</th>
<th>DESTINATION FIELD</th>
<th>ISSUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>msdyn_serviceaccount.accountnumber</td>
<td>=</td>
<td>ORDERINGCUSTOMERACCOUNTNUMBER [ORDERINGCUSTOMERACCOUNTNUMBER]</td>
<td></td>
</tr>
<tr>
<td>msdyn_city</td>
<td>=</td>
<td>DELIVERYADDRESSCITY [DELIVERYADDRESSCITY]</td>
<td></td>
</tr>
<tr>
<td>msdyn_country</td>
<td>=</td>
<td>DELIVERYADDRESSCOUNTRYREGIONECODE [DELIVERYADDRESSCOUNTRYREGIONECODE]</td>
<td></td>
</tr>
<tr>
<td>transactioncurrencyid.iso currency code</td>
<td>=</td>
<td>CURRENTCODE [CURRENTCODE]</td>
<td></td>
</tr>
<tr>
<td>msdyn_address1</td>
<td>=</td>
<td>DELIVERYADDRESSSTREET [DELIVERYADDRESSSTREET]</td>
<td></td>
</tr>
<tr>
<td>msdyn_name</td>
<td>=</td>
<td>SALESORDERNUMBER [SALESORDERNUMBER]</td>
<td></td>
</tr>
<tr>
<td>msdyn_postalcode</td>
<td>=</td>
<td>DELIVERYADDRESSZIPCODE [DELIVERYADDRESSZIPCODE]</td>
<td></td>
</tr>
<tr>
<td>msdyn_address2</td>
<td>=</td>
<td>DELIVERYADDRESSSTREETNUMBER [DELIVERYADDRESSSTREETNUMBER]</td>
<td></td>
</tr>
<tr>
<td>msdyn_stateorprovince</td>
<td>=</td>
<td>DELIVERYADDRESSSTATE [DELIVERYADDRESSSTATE]</td>
<td></td>
</tr>
<tr>
<td>msdyn_billingaccount.accountnumber</td>
<td>=</td>
<td>INVOICECUSTOMERACCOUNTNUMBER [INVOICECUSTOMERACCOUNTNUMBER]</td>
<td></td>
</tr>
<tr>
<td>DeliveryAddressName</td>
<td>=</td>
<td>DELIVERYADDRESSNAME [DELIVERYADDRESSNAME]</td>
<td></td>
</tr>
<tr>
<td>IsDeliveryAddressOrderSpecific</td>
<td>=</td>
<td>ISDELIVERYADDRESSORDERSPECIFIC [ISDELIVERYADDRESSORDERSPECIFIC]</td>
<td></td>
</tr>
<tr>
<td>DeliveryAddressDescription</td>
<td>=</td>
<td>DELIVERYADDRESSDESCRIPTION [DELIVERYADDRESSDESCRIPTION]</td>
<td></td>
</tr>
<tr>
<td>SalesOrderOriginType</td>
<td>=</td>
<td>SALESORDERORIGINTYPE [SALESORDERORIGINTYPE]</td>
<td></td>
</tr>
<tr>
<td>msdyn_systemstatus</td>
<td>=</td>
<td>EXTERNALWORKORDERSTATUS [EXTERNALWORKORDERSTATUS]</td>
<td></td>
</tr>
</tbody>
</table>

**Work Orders with Project (Field Service to Supply Chain Management): WorkOrderHeaderProject**
### Work Orders with Project (Field Service to Supply Chain Management): WorkOrderProduct

<table>
<thead>
<tr>
<th>Source Field</th>
<th>MAP Type</th>
<th>Destination Field</th>
<th>Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>msdyn_city</td>
<td>=</td>
<td>msdyn_city</td>
<td>DeliveryAddressCity</td>
</tr>
<tr>
<td>msdyn_country</td>
<td>=</td>
<td>msdyn_country</td>
<td>DeliveryAddressCountryRegionIsocode</td>
</tr>
<tr>
<td>msdyn_address1</td>
<td>=</td>
<td>msdyn_address1</td>
<td>DeliveryAddressStreet</td>
</tr>
<tr>
<td>msdyn_name</td>
<td>=</td>
<td>msdyn_name</td>
<td>SalesOrderNumber [SalesOrderNumber]</td>
</tr>
<tr>
<td>msdyn_postalcode</td>
<td>=</td>
<td>msdyn_postalcode</td>
<td>DeliveryAddressZipCode</td>
</tr>
<tr>
<td>msdyn_address2</td>
<td>=</td>
<td>msdyn_address2</td>
<td>DeliveryAddressStreetNumber</td>
</tr>
<tr>
<td>msdyn_stateprovince</td>
<td>=</td>
<td>msdyn_stateprovince</td>
<td>DeliveryAddressState</td>
</tr>
<tr>
<td>DeliveryAddressName</td>
<td>=</td>
<td>DeliveryAddressName</td>
<td>DeliveryAddressName</td>
</tr>
<tr>
<td>IsDeliveryAddressOrderSpecific</td>
<td>=</td>
<td>IsDeliveryAddressOrderSpecific</td>
<td>IsDeliveryAddressOrderSpecific</td>
</tr>
<tr>
<td>DeliveryAddressDescription</td>
<td>=</td>
<td>DeliveryAddressDescription</td>
<td>DeliveryAddressDescription</td>
</tr>
<tr>
<td>SalesOrderOriginType</td>
<td>=</td>
<td>msdyn_originstatus</td>
<td>ExternalWorkOrderStatus</td>
</tr>
<tr>
<td>msdyn_systemstatus</td>
<td>=</td>
<td>msdyn_systemstatus</td>
<td>ExternalWorkOrderStatus</td>
</tr>
<tr>
<td>msynce_externalproject, msynce_projectnumber, msynce_externallkey</td>
<td>=</td>
<td>msynce_externalproject, msynce_projectnumber, msynce_externallkey</td>
<td>ProjectID (ProjectID)</td>
</tr>
</tbody>
</table>

### Work Orders with Project (Field Service to Supply Chain Management): WorkOrderService

<table>
<thead>
<tr>
<th>Source Field</th>
<th>MAP Type</th>
<th>Destination Field</th>
<th>Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>msdyn_product.productnumber</td>
<td>=</td>
<td>productnumber</td>
<td>ProductNumber [ProductNumber]</td>
</tr>
<tr>
<td>OrderedSalesQuantity</td>
<td>=</td>
<td>ordersalesquantity</td>
<td>OrderedSalesQuantity [OrderedSalesQuantity]</td>
</tr>
<tr>
<td>msdyn_workorder.msdyn_name</td>
<td>=</td>
<td>msdyn_workorder.msdyn_name</td>
<td>SalesOrderNumber [SalesOrderNumber]</td>
</tr>
<tr>
<td>msdyn_lineorder</td>
<td>=</td>
<td>msdyn_lineorder</td>
<td>LineCreationSequenceNumber [LineCreationSequenceNumber]</td>
</tr>
<tr>
<td>msdyn_unitname</td>
<td>=</td>
<td>msdyn_unitname</td>
<td>SalesUnitSymbol [SalesUnitSymbol]</td>
</tr>
<tr>
<td>SalesPrice</td>
<td>=</td>
<td>salesprice</td>
<td>SalesPrice [SalesPrice]</td>
</tr>
<tr>
<td>TotalDiscountAmount</td>
<td>=</td>
<td>totaldiscountamount</td>
<td>TotalDiscountAmount [TotalDiscountAmount]</td>
</tr>
<tr>
<td>PickedQuantity</td>
<td>=</td>
<td>pickedquantity</td>
<td>PickedQuantity [PickedQuantity]</td>
</tr>
<tr>
<td>WarehouseName</td>
<td>=</td>
<td>shippingwarehouse</td>
<td>ShippingWarehouse [ShippingWarehouse]</td>
</tr>
<tr>
<td>Source Field</td>
<td>Map Type</td>
<td>Destination Field</td>
<td>Issues</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>----------</td>
<td>--------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>msdyn_service.productnumber</td>
<td>=</td>
<td>PRODUCTNUMBER [PRODUCTNUMBER]</td>
<td></td>
</tr>
<tr>
<td>DurationHours [DurationHours]</td>
<td>=</td>
<td>ORDEREDSALESCOUNTY [ORDEREDSALESCOUNTY]</td>
<td></td>
</tr>
<tr>
<td>msdyn_workorder.msdyn.name</td>
<td>=</td>
<td>SALESORDERNUMBER [SALESORDERNUMBER]</td>
<td></td>
</tr>
<tr>
<td>msdyncc_externallineorder</td>
<td>=</td>
<td>LINCREATIONSSEQUENCENUMBER [LINCREATIONSSEQUENCE]</td>
<td></td>
</tr>
<tr>
<td>msdyn_unit.name</td>
<td>Fm</td>
<td>SALESUNITSYMBO [SALESUNITSYMBO]</td>
<td></td>
</tr>
<tr>
<td>SalesPrice [SalesPrice]</td>
<td>=</td>
<td>SALESPRICE [SALESPRICE]</td>
<td></td>
</tr>
<tr>
<td>TotalDiscountAmount [TotalDiscountAmount]</td>
<td>=</td>
<td>TOTALDISCOUNTAMOUNT [TOTALDISCOUNTAMOUNT]</td>
<td></td>
</tr>
</tbody>
</table>
You can consume web services by adding new class libraries. In Microsoft Dynamics AX 2012, you could consume web services from X++ code by adding Microsoft Visual Studio projects as a reference and by using Aif::CreateServiceClient. This scenario is supported, but the steps have changed. Application Integration Framework (AIF) is no longer supported.

The following steps show how to consume an external StockQuote service from X++.

Note that the web service URL in this sample is fictional. There is no known web service at http://www.contoso.net/stockquote.asmx. To make this code work you will need to adapt it to your specific web service.

1. Create a new Class Library project in Visual Studio, and name it ExternalServiceLibrary.csproj.

2. In the Visual Studio project, add a service reference to the external web service:
   http://www.contoso.net/stockquote.asmx.

3. Create a new static class, and wrap the StockQuote service operation as shown in the following example.

   ```csharp
   public static string GetQuote(string s)
   {
       var binding = new System.ServiceModel.BasicHttpBinding();
       var endpointAddress = new EndpointAddress("http://www.contoso.net/stockquote.asmx");
       ServiceLibrary.QuoteReference.StockQuoteSoapClient client = new
       ServiceLibrary.QuoteReference.StockQuoteSoapClient(binding, endpointAddress);

       //GetQuote is the operation on the StockQuote service
       return client.GetQuote("MSFT");
   }
   ``

4. Build the project. The binary ExternalServiceLibrary.dll is created.

5. Create a new Dynamics project in Visual Studio.

6. Add ExternalServiceLibrary.dll as a reference.

7. In the X++ class, you can use the external web services that were referenced in ExternalServiceLibrary.dll.

   ```csharp
   public static void main(Args _args)
   {
       info(ServiceLibrary.StockQuoteClass::GetQuote("MSFT"));
   }
   ```
This topic provides overview and the setup information for the **Electronic messages** (EM) functionality.

Recently, the governments and legislative authorities of various countries and regions around the world have implemented reporting requirements for companies that are registered in those countries or regions. The purpose of the requirements is to enable data to be obtained from those companies in electronic format, directly from the systems where it was accounted, stored, and processed.

The EM functionality in Microsoft Dynamics 365 Finance supports various processes for electronic interoperation between Finance and the systems that governments and legislative authorities offer for reporting, submitting, and receiving official information.

The EM functionality is integrated with the **Electronic Reporting** (ER) module. You can set up ER formats for electronic messages. For more information, see [Electronic reporting (ER)](#).

### Basic concepts for EM functionality

The EM functionality is based on the following entities:

- **Electronic message** – A report or declaration that should be reported or transmitted internally, such as a report that is sent to a tax office.
- **Electronic message items** – Records that should be included in the message that is reported.
- **Electronic message processing** – A chain of actions that should be run to collect the required data, generate reports, store data in Azure Blob storage, transmit reports outside the system, receive responses from outside the system, and, based on the information that is received, update the database. The actions in the chain can be linked or unlinked.

The following illustration shows the flow of data for EM.

### Scenarios supported by the EM functionality

The EM functionality supports the following scenarios:

- Manually create messages and generate reports that are based on associated exporting ER formats of various types. These types include Microsoft Excel, XML, JavaScript Object Notation (JSON), PDF, text, and Microsoft Word.
- Automatically create and process messages that are based on information that was requested and received...
Security privileges

The following security privileges are available for electronic messages.

<table>
<thead>
<tr>
<th>SECURITY PRIVILEGE</th>
<th>ACCESS LEVEL</th>
<th>ASSOCIATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintain electronic messages</td>
<td>This privilege gives full access to the EM functionality. If you have this</td>
<td>This privilege is included in the Maintain sales tax transactions security duty. That duty, in turn, is included in the Accountant security role.</td>
</tr>
<tr>
<td></td>
<td>privilege, you can set up electronic messaging and run all the processing.</td>
<td></td>
</tr>
<tr>
<td>View electronic messages</td>
<td>This privilege gives read-only access to the EM functionality. If you have this</td>
<td>This privilege is included in the Inquire into sales tax transaction status security duty. That duty, in turn, is included in the following security roles:</td>
</tr>
<tr>
<td></td>
<td>privilege, you can view the electronic messaging settings and messages.</td>
<td>• Collections manager</td>
</tr>
<tr>
<td></td>
<td>However, you can't set up or run anything.</td>
<td>• Accounts receivable clerk</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Accounts receivable manager</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Tax accountant</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Accountant</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Accounting manager</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Accounting supervisor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Sales manager</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Accounts payable clerk</td>
</tr>
<tr>
<td>Operate electronic messages</td>
<td>This privilege gives access only to the Electronic messages and Electronic</td>
<td>This privilege is included in the Operate electronic messages security duty. That duty, in turn, is included in the Electronic messages operator security role.</td>
</tr>
<tr>
<td></td>
<td>message items pages. If you have this privilege, you can run all the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>processing that is called from those pages.</td>
<td></td>
</tr>
</tbody>
</table>

Country-specific regulatory features supported by the EM functionality

The following table provides information about some country-specific regulatory features that are supported by the EM functionality.

- Collect and process information from a data source as message items. The data source is a Finance table.
- Store additional information, and evaluate various values by calling specifically defined executable classes in relation to messages or message items.
- Aggregate information that is collected in message items, split that information by message, and generate reports that are in associated exporting ER formats.
- Transmit the reports that are generated to a web service by using security information that is stored in the Azure key vault.
- Receive a response from a web service, interpret the response, and update data in Finance as appropriate.
- Store and review all the reports that are generated.
- Store and review all the log information that is related to actions that are run for a message or message item.
- Control the processing through various message statuses and message item statuses.
- From an authority by using an associated importing ER format.
- Collect and process information from a data source as message items. The data source is a Finance table.
- Store additional information, and evaluate various values by calling specifically defined executable classes in relation to messages or message items.
- Aggregate information that is collected in message items, split that information by message, and generate reports that are in associated exporting ER formats.
- Transmit the reports that are generated to a web service by using security information that is stored in the Azure key vault.
- Receive a response from a web service, interpret the response, and update data in Finance as appropriate.
- Store and review all the reports that are generated.
- Store and review all the log information that is related to actions that are run for a message or message item.
- Control the processing through various message statuses and message item statuses.

The following table provides information about some country-specific regulatory features that are supported by the EM functionality.
<table>
<thead>
<tr>
<th>COUNTRY</th>
<th>FEATURE NAME</th>
<th>FEATURE DEMO RECORDING</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spain</td>
<td>Immediate Supply of Information on VAT (Suministro Inmediato de Información del IVA, SII)</td>
<td></td>
</tr>
<tr>
<td>Hungary</td>
<td>Online invoicing system</td>
<td></td>
</tr>
<tr>
<td>United Kingdom</td>
<td>Making Tax Digital (MTD) – VAT statement submission</td>
<td>Finance and Operations: UK Digital Tax - VAT Declaration In Dynamics 365</td>
</tr>
<tr>
<td>Lithuania</td>
<td>i.SAF reporting</td>
<td></td>
</tr>
<tr>
<td>Poland</td>
<td>VAT declaration with registers (JPK_V7M, VDEK)</td>
<td>Dynamics 365 Finance: SAF/JPK VAT Audit Registers</td>
</tr>
<tr>
<td>Netherlands</td>
<td>VAT declaration for Netherlands</td>
<td></td>
</tr>
<tr>
<td>Czech Republic</td>
<td>VAT declaration</td>
<td></td>
</tr>
<tr>
<td>Brazil</td>
<td>SPED-Reinf</td>
<td></td>
</tr>
<tr>
<td>Russia</td>
<td>VAT declaration</td>
<td></td>
</tr>
<tr>
<td>Russia</td>
<td>Accounting reporting in electronic format</td>
<td></td>
</tr>
<tr>
<td>Russia</td>
<td>Profit tax declaration</td>
<td></td>
</tr>
<tr>
<td>Russia</td>
<td>Assessed tax declaration</td>
<td></td>
</tr>
<tr>
<td>Russia</td>
<td>Transport tax declaration</td>
<td></td>
</tr>
<tr>
<td>Russia</td>
<td>Land tax declaration</td>
<td></td>
</tr>
</tbody>
</table>
This topic describes how you can use the data management framework to manage data entities and data entity packages in Finance and Operations.

The data management framework consists of the following concepts:

- **Data entities** - A data entity is a conceptual abstraction and encapsulation of one or more underlying tables. A data entity represents a common data concept or functionality, for example, Customers or Vendors. Data entities are intended to be easily understood by users familiar with business concepts. After data entities are created, you can reuse them through the Excel Add-in, use them to define import/export packages, or use them for integrations.

- **Data project** - A project that contains configured data entities, which include mapping and default processing options.

- **Data job** - A job that contains an execution instance of the data project, uploaded files, schedule (recurrence), and processing options.

- **Job history** - Histories of source to staging and staging to target jobs.

- **Data package** - A single compressed file that contains a data project manifest and data files. This is generated from a data job and used for import or export of multiple files with the manifest.

The data management framework supports using data entities in the following core data management scenarios:

- Data migration
- Set up and copy configurations
- Integration

### Data entities

Data entities provide conceptual abstraction and encapsulation of underlying table schema that represent data concepts and functionalities. In Microsoft Dynamics AX 2012, most tables, like the Customer and Vendor tables, were de-normalized and split into multiple tables. This was beneficial from a database design point of view, but made it difficult for implementers and ISV’s to use without a thorough understanding of the physical schema.

Data entities were introduced as part of data management to be used as a layer of abstraction to easily understand by using business concepts. In previous versions there were multiple ways to manage data, such as Microsoft Excel Add-ins, AIF, and DIXF. The concept of data entities combines those different concepts into one. After data entities are created, you should be able to reuse them for an Excel Add-ins, import/export, or integration. The following table shows core data management scenarios.

<table>
<thead>
<tr>
<th>Data Migration</th>
<th>Setup and copy configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Migrate reference, master, and document data from legacy or external systems.</td>
<td>- Copy configuration between company/environments.</td>
</tr>
<tr>
<td></td>
<td>- Configure processes or modules using the Lifecycle Services (LCS) environment.</td>
</tr>
</tbody>
</table>
Integration

- Real-time service based integration.
- Asynchronous integration.

Data migration

Using the data management framework, you can quickly migrate reference, master, and document data from legacy or external systems. The framework is intended to help you quickly migrate data by using the following features:

- You can select only the entities you need to migrate.
- If an import error occurs, you can skip selected records and choose to proceed with the import using only the good data, opting to then fix and import the bad data later. You will be able to partially continue and use errors to quickly find bad data.
- You can move data entities straight from one system to another, without having to go through Excel, or XML.
- Data imports can be easily scheduled using a batch, which offers flexibility when it is required to run. For example, you can migrate customer groups, customers, vendors, and other data entities in the system at any time.

Set up and copy configuration

You can use the data management framework to copy configurations between companies or environments, and configure processes or modules using Microsoft Dynamics Lifecycle Services (LCS).

Copying configurations is intended to make it easier to start a new implementation, even if your team doesn’t deeply understand the structure of data that needs to be entered, or data dependencies, or which sequence to add data to an implementation.

The data management framework allows you to:

- Move data between two similar systems
- Discover entities and dependencies between entities for a given business process or module
- Maintain a reusable library of data templates and datasets
- Use data packages to create incremental data entities. Data entities can be sequenced inside the packages. You can name data packages, which can be easily identifiable during import or export. When building data packages, data entities can be mapped to staging tables in grids or by using a visual mapping tool. You can also drag-and-drop columns manually.
- View data during imports, so you can compare data, and ensure that it is valid.

Working with data entities

The following sections provide quick snapshots of the different functionalities of data management using data entities. The goal is to help you strategize and make effective decisions on how to best utilize the available tools during data migration. You will also find tips and tricks on how to effectively use each area during data migration. A list of available data entities for each area can also be found with the suggested data sequences, showing data dependencies. Microsoft provides data packages that can be found on Lifecycle Services (LCS) as an initial guide. The information in this document can be used as a guide for creating your own packages. The description of each data entity shows what the object contains and if it is needed during data migration.

Sequencing

There are two types of sequencing that should be considered when working with data entities.

- Sequencing data entities within a data package
- Sequencing the order of data package imports
Sequence data entities within a data package

1. When a user adds data entities to a data project, by default, a sequence is set for the order in which the entities will load. The first entity added to the project will be set as the first entity to load, the next entity added will be second, the next entity will be third, and so on.

   For example, if a user added two entities in this order, **Sales tax codes** and **Sales Tax groups**, then **Sales tax codes** is assigned an entity sequence of 1.1.1, and **Sales tax groups** is assigned an entity sequence of 1.1.2. The sequence level indicates that the second entity will not start the import process until the first level is finished.

2. To view or edit a sequence, click the **Entity sequence** button on the Action Pane of the data project.

3. In the Definition group entity sequence, you can see the execution units and the sequence. You can change sequence by selecting the data entity in the list, setting a different Execution unit or Sequence in level, and then clicking **Update selected**. After clicking **Update selected**, the entity will move up or down in the entity list.

Example

The following screenshot shows the entity sequence that is set for the Sales Tax CodeGroups data package.

<table>
<thead>
<tr>
<th>Entity</th>
<th>Execution unit</th>
<th>Level in execution unit</th>
<th>Sequence in level</th>
<th>Fail level on error</th>
<th>Fail execution unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales tax codes</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales tax code values</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales tax code limits</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales tax groups</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales tax group details</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales tax item groups</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales tax exempt numbers</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales tax exempt code</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales tax reporting codes</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In order to successfully import sales tax codes and groups, the sales tax codes and details have to be loaded first, before sales tax groups can be imported. Sales tax codes and groups are all in Execution unit = 1, but the sequences are in the order that they will be imported. Other related sales tax entities that are not dependent upon other data entities being loaded are included in the package. For example, sales tax exempt numbers is set in its own Execution unit = 2. This data entity will start loading immediately because there are no dependencies on other entities loading before it.

Sequence data package imports

In order to successfully load data, it's important to set the correct order for importing data packages, because of dependencies that exist within and across modules. The numbering format that has been created for the data packages within LCS are as follows:

- First segment: Module
- Second segment: Data type (setup, master, transaction)
- Third segment: Sequence number
The following tables provide more information about the default numbering format.

### Module numbers

<table>
<thead>
<tr>
<th>Module</th>
<th>Module Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>System administration</td>
<td>01</td>
</tr>
<tr>
<td>General ledger</td>
<td>03</td>
</tr>
<tr>
<td>Public Sector</td>
<td>04</td>
</tr>
<tr>
<td>HRM</td>
<td>05</td>
</tr>
<tr>
<td>Accounts payable</td>
<td>10</td>
</tr>
<tr>
<td>Accounts receivable</td>
<td>11</td>
</tr>
<tr>
<td>Budgeting</td>
<td>12</td>
</tr>
<tr>
<td>Cash and bank management</td>
<td>13</td>
</tr>
<tr>
<td>Compliance and internal controls</td>
<td>14</td>
</tr>
<tr>
<td>Cost accounting</td>
<td>15</td>
</tr>
<tr>
<td>Fixed assets</td>
<td>16</td>
</tr>
<tr>
<td>Inventory management</td>
<td>19</td>
</tr>
<tr>
<td>Master planning</td>
<td>20</td>
</tr>
<tr>
<td>Organization administration</td>
<td>21</td>
</tr>
<tr>
<td>Payroll</td>
<td>22</td>
</tr>
<tr>
<td>Procurement and sourcing</td>
<td>23</td>
</tr>
<tr>
<td>Product information management</td>
<td>24</td>
</tr>
<tr>
<td>Production control</td>
<td>25</td>
</tr>
<tr>
<td>Project management and accounting</td>
<td>26</td>
</tr>
<tr>
<td>Retail</td>
<td>27</td>
</tr>
<tr>
<td>Sales and marketing</td>
<td>28</td>
</tr>
<tr>
<td>Service management</td>
<td>29</td>
</tr>
<tr>
<td>Trade allowance management</td>
<td>31</td>
</tr>
<tr>
<td>Transportation management</td>
<td>32</td>
</tr>
<tr>
<td>Travel and expense</td>
<td>33</td>
</tr>
<tr>
<td>Warehouse management</td>
<td>34</td>
</tr>
</tbody>
</table>

### Data type numbers

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Data Type Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup</td>
<td>1</td>
</tr>
<tr>
<td>Master</td>
<td>4</td>
</tr>
<tr>
<td>Transaction</td>
<td>8</td>
</tr>
</tbody>
</table>

### Sequence number

<table>
<thead>
<tr>
<th>Numbering Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module #. Data Type Reference .001 (Sequence Number)</td>
</tr>
<tr>
<td>01.1.001 System</td>
</tr>
<tr>
<td>01.1.002 System</td>
</tr>
<tr>
<td>01.4.001 System</td>
</tr>
<tr>
<td>01.4.002 System</td>
</tr>
<tr>
<td>03.1.001 General Ledger</td>
</tr>
</tbody>
</table>

Data packages follow the sequence number, followed by the module abbreviation, and then a description. The following example shows General ledger data packages:

1. 03.1.001 GL - Exchange Rates
2. 03.1.002 GL - Chart of Accounts
3. 03.1.003 GL - Account Structures
4. 03.1.004 GL - Fiscal Calendar
5. 03.1.005 GL - Ledger Setup
6. 03.1.006 GL - Ledger Journals
7. 03.1.007 GL - Allocations

### Mapping

When working with data entities, mapping an entity to a source is automatic. The automatic mapping of fields
can be overridden if needed.

**View mapping**

To view how an entity is mapped, locate the tile for the entity in the project, and then click **View map**.

We provide mapping visualization view (default) and mapping details view. A red asterisk (*) identifies any required fields in an entity. These fields must be mapped in order to work with the entity. Other fields can be unmapped as required when working with the entity.

- To unmap a field, highlight the field in either column (**Entity** or **Source**), click **Delete selection**, and then click **Save**. After saving, close the form to return to the project.

The field mapping from source to staging can also be edited after import using the same process.

**CURRENCIES : CURRENCIES**

**Map source to staging**

**Regenerate a map**

If you have extended an entity (added fields) or if the automatic mapping appears to be incorrect, the mapping of the entity can be regenerated in the **Mapping** form.

1. To do this, click **Generate source mapping**.

A message will display asking, “Do you want to generate the mapping from scratch?”

2. Click **Yes** to regenerate the mapping.

**Generate data**

If you have fields in entities that you want the system to generate data for on import, instead of providing the data in the source file, you can use the auto-generated functionality in the mapping details for the entity. For example, if you want to import customers and customer address information, but the address information was not previously imported with the Global Address Book entities, you can have the entity auto-generate the party number upon import and the GAB information will be created. To access this functionality, view the map of the entity and click the **Mapping details** tab. Select the fields that you want to auto-generate. This will change the source field to **Auto**.
Turn off automatically generated number sequences

Many entities support automatic generation of identifiers based on number sequence setup. For example, when creating a product, the product number is automatically generated and the form does not allow you to edit values manually.

It is possible to enable manual assignment of number sequences for a specific entity.
After you have enabled manual assignment, you can provide manually assigned numbers instead.

Export

Export is the process of retrieving data from a system using data entities. The export process is done through a project. When exporting, you have a lot of flexibility as to how the export project is defined. You can choose which data entities to export, but also the number of entities, the file format used (there are 14 different formats to choose for export), and apply a filter to each entity to limit what is exported. After the data entities have been pulled into the project, the sequencing and mapping described earlier can be performed for each export project.

After the project is created and saved you can export the project to create a job. During the export process, you can see a graphical view of the status of the job and the record count. This view shows multiple records so you can review the status of each record prior to downloading the actual files.
After the job is completed you can choose how to download the files: each data entity can be a separate file, or by combining the files into a package. If there are multiple data entities in the job, choosing the package option will speed up the upload process. The package is a zip file, containing a data file for each entity as well as a package header and manifest. These additional documents are used when importing in order to add the data files to the correct data entities and sequence the import process.

**Import**

Import is the process of pulling data into a system using data entities. The import process is done through the Import tile in the Data Management workspace. Data can be imported either for individual entities or for a group of logically related entities that are sequenced in the correct order. The file formats vary depending on the type of import. For an entity, it can be an Excel file that is comma-separated, tab-separated, or text. For a data package, it is a .zip file. In both cases, the files are exported using the above mentioned export process.

**Import a data package**

1. Log into the environment using a login with sufficient privileges (typically this is the Administrator role).
2. On the dashboard, click the Data Management workspace.
3. Click the Import tile.
4. On the next page, do the following:
   a. Provide a name.
   b. In the Source Data Format field, select Package.
   c. Click the Upload button and choose the appropriate package file from the location for the data being imported. This will import all the files from the package.
d. Click Save, and then click Import.

Import multiple data packages

Use one of the following methods to import multiple data packages.

- Create a new job for each package, and then repeat steps 4(a) through 4(d) above, for each package.

- Create one job to import multiple packages in a sequence. Repeat steps 4(a) through 4(c) above, and then repeat step 4(c) for all packages that need to be imported. After you select the packages, execute step 4(d) to import the data from the selected data packages through a single job.

After you click Import, the data will be imported through staging tables. The progress of the import can be tracked using the Refresh button in the upper-right corner of the screen.

Troubleshoot data package processing

This section provides troubleshooting information for the different stages of data package processing.

- Status and error details of a scheduled job can be found under the Job history section in the Data management form.

- Status and error details of previous runs for data entities can be displayed by selecting a data project and clicking Job history. In the Execution history form, select a job, and click View staging data and View execution log. The previous runs include data project runs that were executed as batch jobs or manually.

Export process troubleshooting
- If you get an error during the export process, click **View execution log** and review the log text, staging log details, and Infolog for more information.
- If you get an error during the export process with a note directing you not to skip staging, turn off the **Skip staging** option, and then add the entity. If you are exporting multiple data entities, you can use the **Skip staging** button for individual data entities.

**Import process troubleshooting**

When uploading data entity files:

- If data entities do not display in **Selected files and entities** after you click **Upload** during the import process, wait a few minutes, and then check whether the OLEDB driver is still installed. If not, then reinstall the OLEDB driver. The driver is Microsoft Access Database Engine 2010 Redistributable – AccessDatabaseEngine_x64.exe.
- If data entities display in **Selected Files and Entities** with a warning after you click **Upload** during the import process, verify and fix the mapping of individual data entities by clicking **View map**. Update the mapping and click **Save** for each data entity.

During data entity import:

- If data entities fail (shown with a red X or yellow triangle icon on the data entity tile) after you click **Import**, click **View staging data** on each tile under the **Execution summary** page to review the errors. Sort and scroll through the records with Transfer status = Error to display the errors in the Message section. Download the staging table. Fix a record (or all records) directly in staging by clicking **Edit, Validate all, and Copy data to target**, or fix the import file (not staging file) and reimport the data.
- If data entities fail (shown with a red x or yellow triangle icon on the data entity tile) after you click **Import**, and **View staging data** shows no data, go back to the **Execution summary** page. Go to **View execution log**, select the data entity, and review the **Log text, Staging log details, and Infolog** for more information. **Staging log details** will display **Error column** (field) details and **Log description** will describe errors in detail.
- If data entities fail, you can check the import file to see if there is an extra line in the file with text which displays, "This is a string that is inserted into Excel as a dummy cell to make the column to support more than 255 characters. By default, an Excel destination component will not support more than 255 characters. The default type of Excel will be set based on the first few rows". This line is added during data export. If this line exists, delete this line, re-package the data entity, and try to import.

**Features flighted in data management and enabling flighted features**

The following features are enabled via flighting. **Flighting** is a concept that allows a feature to be ON or OFF by default.

<table>
<thead>
<tr>
<th>FLIGHT NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMFEnableAllCompanyExport</td>
<td>Enables BYOD export from all companies in the same export job (supported for BYOD only and not files). By default, this is OFF. This flight is no longer needed after Platform update 2.7 because this feature can be turned ON using a parameter in data management framework parameters.</td>
</tr>
<tr>
<td>DMFExportToPackageForceSync</td>
<td>Enables synchronous execution of data package API export. By default, it’s asynchronous.</td>
</tr>
<tr>
<td>EntityNamesInPascalCaseInXMLFiles</td>
<td>Enables behavior where entity names are in Pascal Case in the XML files for entities. By default, the names are in upper case.</td>
</tr>
<tr>
<td>FLIGHT NAME</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DMFByodMissingDelete</td>
<td>Enables the old behavior where under certain conditions, certain delete operations were not synced to BYOD using change tracking.</td>
</tr>
<tr>
<td>DMFDisableExportFieldsMappingCache</td>
<td>Disables caching logic when building target field mapping.</td>
</tr>
<tr>
<td>EnableAttachmentForPackageApi</td>
<td>Enables attachments functionality in the package API.</td>
</tr>
<tr>
<td>FailErrorOnBatchForExport</td>
<td>Enables fail on error at execution unit or level for export jobs.</td>
</tr>
<tr>
<td>IgnorePreventUploadWhenZeroRecord</td>
<td>Disables ‘prevent upload when zero records’ functionality.</td>
</tr>
<tr>
<td>DMFInsertStagingLogToContainer</td>
<td>By default this is ON. This improves performance and functional issues with error logs in the staging table.</td>
</tr>
<tr>
<td>ExportWhileDataEntityListIsBeingRefreshed</td>
<td>When enabled, additional validations are made on mappings when a job is scheduled while entity refresh is in progress. By default, this is OFF.</td>
</tr>
<tr>
<td>DMFDisableXSLTTransformationForCompositeEntity</td>
<td>This can disable the application of transformations on composite entities.</td>
</tr>
<tr>
<td>DMFDisableInputFileCheckInPackageImport</td>
<td>Additional validations are made to ensure if any entity file is missing from a data package, error message is shown. This is the default behavior. If required, this can be turned OFF by this flight.</td>
</tr>
<tr>
<td>FillEmptyXMLFileWhenExportingCompositeEntity</td>
<td>Prior to Platform update 15, when exporting composite entities that did not have any records to export, the XML file generated did not have any schema elements. This behavior can be changed to output empty schema by enabling this flight. By default, the behavior will still be to output empty schema.</td>
</tr>
<tr>
<td>EnableNewNamingForPackageAPIExport</td>
<td>A fix was made to ensure unique names are used for the execution ID when ExportToPackage is used for export scenarios. Duplicate execution ID's were being created when ExportToPackage was called in quick succession. To preserve compatibility, this behavior is OFF by default. Turning this flight ON will enable this new behavior where new naming convention for execution ID's will ensure unique names.</td>
</tr>
<tr>
<td>DMFDisableDoubleByteCharacterExport</td>
<td>A fix was made to ensure that data can be exported when the format is configured to use code page 932 setting. If an issue is encountered in relation to double byte exports, this fix can be turned off by disabling this flight to unblock, if applicable.</td>
</tr>
<tr>
<td>DisablePendingRecordFromJobStatus</td>
<td>A fix was made to ensure that pending records are taken into consideration while evaluating the final status of an import job. If implementations have a dependency on the status evaluation logic and this change is considered a breaking change for an implementation, this new logic can be disabled using this flight.</td>
</tr>
<tr>
<td>FLIGHT NAME</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DMFDisableEnumFieldDefaultValueMapping</td>
<td>A fix was made to ensure that default values set in advanced mapping for enum fields are successfully saved in the data package manifest file when generating the data package. This makes it possible for the data package to be used as a template for integrations when such advanced mappings are used. This fix is protected by this flight and can be disabled if the previous behavior is still needed (which is to always set the value to 0 in the data package manifest).</td>
</tr>
<tr>
<td>DMFDisableEnumFieldDefaultValueMapping</td>
<td>This flight only applies to Platform update 34 and non-production environments. A fix was made in Platform update 34 to prevent scripting in XSLT. However, this resulted in breaking some functionality that was dependent on scripting. As a result, this flight has been turned ON by Microsoft in all production environments as a preventive measure. In non-production environments, this must be added by customers if they encounter XSLT failures related to scripting. From Platform update 36 onward, a code change was made to revert the Platform update 34 change so this flight does not apply from Platform update 36 onward. Even if you enable this flight in Platform update 34, upgrading to Platform update 35 will not cause any negative impact due to this flight being ON from Platform update 34.</td>
</tr>
<tr>
<td>DMFExecuteSSISInProc</td>
<td>This flight is OFF by default. This is related to a code fix that was made to run SQL Server Integration Services (SSIS) out of in-process to optimize memory utilization of SSIS when running DIXF jobs. However, this change has caused a regression in a scenario where if the DIXF data project name has an apostrophe (‘) in it, then the job will fail with an error. If you encounter this issue, removing the (‘) in the data project name will resolve the failure. However, if for some reason the name cannot be changed, then this flight can be enabled to overcome this error. Enabling this flight will make SSIS run in-process as before, which could lead to higher memory consumption when running DIXF jobs.</td>
</tr>
</tbody>
</table>

The following steps enable a flight in a non-production environment. Execute the following SQL command.

For enabling flights in a production environment, a support case must be logged with Microsoft:

- After running the SQL statement, ensure that the following is set in the web.config file on each of the AOS's. add key="DataAccess.FlightingServiceCatalogID" value="12719367"

- After making the above change, perform an IISReset on all AOS's.

```sql
INSERT INTO SYSFLIGHTING
([FLIGHTNAME], [ENABLED], [FLIGHTSERVICEID], [PARTITION], [RECID], [RECVERSION])
VALUES ('name', 1, 12719367, PARTITION, RECID, 1)
```

- Partition - Partition ID from the environment, which can be obtained by querying (select) for any record. Every record will have a partition ID that must be copied and used here.
- RecID - Same ID as partition. However, if multiple flights are enabled, then this can be partition ID + 'n' to ensure it has a unique value
- RecVersion = 1

Additional resources
- Data entities overview
This topic defines and provides an overview of data entities. It includes information about the capabilities of data entities, the scenarios that they support, the categories that are used for them, and the methods for creating them.

Overview

A **data entity** is an abstraction from the physical implementation of database tables. For example, in normalized tables, a lot of the data for each customer might be stored in a customer table, and then the rest might be spread across a small set of related tables. In this case, the data entity for the customer concept appears as one de-normalized view, in which each row contains all the data from the customer table and its related tables. A data entity encapsulates a business concept into a format that makes development and integration easier. The abstracted nature of a data entity can simplify application development and customization. Later, the abstraction also insulates application code from the inevitable churn of the physical tables between versions. **To summarize:** Data entity provides conceptual **abstraction** and **encapsulation** (de-normalized view) of underlying table schemas to represent key data concepts and functionalities.

**Capabilities**

A data entity has the following capabilities:

- It replaces diverging and fragmented concepts of AXD, Data Import/Export Framework (DIXF) entities, and aggregate queries with single concept.
- It provides a single stack to capture business logic, and to enable scenarios such as import/export, integration, and programmability.
- It becomes the primary mechanism for exporting and importing data packages for Application Lifecycle Management (ALM) and demo data scenarios.
- It can be exposed as OData services, and then used in tabular-style synchronous integration scenarios and Microsoft Office integrations.
A consumer wants to access data that is related to a customer object, but this data is currently scattered across multiple normalized tables, such as DirParty, CustTable, LogisticPostalAddress, and LogisticElectronicAddress. Therefore, the process of reading and writing customer data is very tedious. Instead, the following customer entity can be designed to encapsulate the entire underlying physical schema into a single de-normalized view. This enables simpler read/write operations and also enables abstraction of any internal interaction between the tables.
Supported scenarios

Data entities support all the following scenarios.

Integration scenarios

Synchronous service (OData)

Data entities enable public application programming interfaces (APIs) on entities to be exposed, which enables synchronous services. Synchronous services are used for the following purposes:

- Office integration
- Third-party mobile apps

Asynchronous integration

Data entities also support asynchronous integration through a data management pipeline. This enables asynchronous and high-performing data insertion and extraction scenarios. Here are some examples:

- Interactive file-based import/export
- Recurring integrations (file, queue, and so on)

Business intelligence

- Aggregate data
- Standardized key performance indicators (KPIs)

Application Lifecycle Management

Besides integration and business intelligence (BI) scenarios, data entities also initially support two critical ALM scenarios. The following two progressive levels of an ALM scenario show the scope of coverage by data entities.
A system implementer will use both a guided data collection wizard and bulk data input mechanisms to **bootstrap the initial deployment** (or module) with configuration data through Microsoft Dynamics Lifecycle Services (LCS). Configuration primarily targets to cover the following entity categories:

- All of Parameter
- Reference
- System parameter
- Number sequence
- Currency

**Data migration from legacy or external systems**

After the initial deployment is up and running, the system implementer will **migrate existing data assets of the customer** into the application, especially the following assets:

- Master data (for example, customers and vendors)
- Subsets of documents (for example, sales orders)

### Categories of entities

Entities are categorized based on their functions and the type of data that they serve. The following are five categories for data entities.

**Parameter**

- Functional or behavioral parameters.
- Required to set up a deployment or a module for a specific build or customer.
- Can include data that is specific to an industry or business. The data can also apply to a broader set of customers.
- Tables that contain only one record, where the columns are values for settings. Examples of such tables exist for Account payable (AP), General ledger (GL), client performance options, workflows, and so on.

**Reference**

- Simple reference data, of small quantity, which is required to operate a business process.
- Data that is specific to an industry or a business process.
- Examples include units, dimensions, and tax codes.

**Master**

- Data assets of the business. Generally, these are the “nouns” of the business, which typically fall into categories such as people, places, and concepts.
- Complex reference data, of large quantity. Examples include customers, vendors, and projects.

**Document**
- Worksheet data that is converted into transactions later.
- Documents that have complex structures, such as several line items for each header record. Examples include sales orders, purchase orders, open balances, and journals.
- The operational data of the business.

**Transaction**
- The operational transaction data of the business.
- Posted transactions. These are non idempotent items such as posted invoiced and balances. Typically, these items are excluded during a full dataset copy to reduce the volume of data that is copied/migrated. Migrating completed transactions can also lead to further complexity in trying to preserve the referential integrity of related data in the new system. In general, transactions from a completed business process are not migrated in detail but in summary.
- Examples include pending invoices.

**Building an entity**

There are multiple ways to create an entity. For example, you can use a wizard, or you can build an entity from a table.

**Building an entity by using a wizard**

The simplest way to build an entity is to use a wizard. This wizard lets you select a root data source and expand to other related data sources, and then select fields for the entity. To start the wizard, add a new item of type **Data entity** to your project. For step-by-step instructions for using the wizard to build an entity, see [Build and consume data entities](#). The following table provides information about the properties that you set for an entity in the wizard.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary data source</td>
<td>The root data source (table or view) that is used to construct the entity. You can add more related data sources, based on this root data source.</td>
</tr>
<tr>
<td>Data entity name</td>
<td>The name of the entity.</td>
</tr>
<tr>
<td>Entity category</td>
<td>The type of entity. Entity categories are similar to table groups for tables. The available categories include <strong>Parameter, Reference, Master, Document, and Transaction</strong>.</td>
</tr>
<tr>
<td>Public entity name</td>
<td>The public resource name for the entity.</td>
</tr>
<tr>
<td>Public collection name</td>
<td>The public resource set name.</td>
</tr>
<tr>
<td>Enable public API</td>
<td>Select this option to enable the entity for OData services.</td>
</tr>
<tr>
<td>Enable data management capabilities</td>
<td>Select this option to enable the entity for asynchronous integrations such as data import/export and connector integration.</td>
</tr>
</tbody>
</table>
**Staging table**
The name of the staging table that will be generated for the entity. The staging table is used in asynchronous integrations and high-volume scenarios.

### Adding data sources

When you build an entity, you start with a root data source. However, you can add additional data sources. You can either manually add new data sources, or select a surrogate foreign key field in the root data source to automatically expand the required data sources.

### Output

When you complete the wizard, it produces the following items:

- Data entity
- Staging table (optional, if data management was enabled)

### Building an entity from a table

You can quickly create an entity from a table, and then customize the properties, data sources, and fields later. Right-click the table, and then select **Addins > Create data entity**.

### Entity list refresh

Entities in an environment must be refreshed using the following guidelines.

- When a new environment is deployed and the user navigates to the data management workspace, entity list refresh starts automatically.
- When code packages are deployed to an environment where data management has already been used, entity list refresh must be manually started from **Data management > Framework parameters > Entity settings > Refresh entity list**.
- When configuration keys are modified, entity list must be refreshed manually from **Data management > Framework parameters > Entity settings > Refresh entity list**.

Refreshing the entity list ensures all entities are available in the environment and that the entities have the latest metadata.

### Configuration keys and data entities

Before you use data entities to import or export data, we recommend that you first determine the impact of configuration keys on the data entities that you are planning to use.
To learn more about configuration keys, see the License codes and configuration keys report.

**Configuration key assignments**

Configuration keys can be assigned to one or all of the following artifacts.

- Data entities
- Tables used as data sources
- Table fields
- Data entity fields

The following table summarizes how configuration key values, on the different artifacts that underlie an object, change the expected behavior of the object.

<table>
<thead>
<tr>
<th>CONFIGURATION KEY SETTING ON DATA ENTITY</th>
<th>CONFIGURATION KEY SETTING ON TABLE</th>
<th>CONFIGURATION KEY SETTING ON TABLE FIELD</th>
<th>CONFIGURATION KEY ON DATA ENTITY FIELD</th>
<th>EXPECTED BEHAVIOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disabled</td>
<td>Not evaluated</td>
<td>Not evaluated</td>
<td>Not evaluated</td>
<td>If the configuration key for the data entity is disabled, the data entity will not be functional. It does not matter whether the configuration keys in the underlying tables and fields are enabled or disabled.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Disabled</td>
<td>Not evaluated</td>
<td>Not evaluated</td>
<td>If the configuration key for a data entity is enabled, the data management framework checks the configuration key on each of the underlying tables. If the configuration key for a table is disabled, that table will not be available in the data entity for functional use. If a table’s configuration key is disabled, the table and data entity configuration key settings are not evaluated. If the primary table in the entity has its configuration key disabled, then the system will act as though the entity’s configuration key were disabled.</td>
</tr>
<tr>
<td>CONFIGURATION KEY SETTING ON DATA ENTITY</td>
<td>CONFIGURATION KEY SETTING ON TABLE</td>
<td>CONFIGURATION KEY SETTING ON TABLE FIELD</td>
<td>CONFIGURATION KEY ON DATA ENTITY FIELD</td>
<td>EXPECTED BEHAVIOR</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>------------------------------------</td>
<td>------------------------------------------</td>
<td>----------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Enabled</td>
<td>Enabled</td>
<td>Disabled</td>
<td>Not evaluated</td>
<td>If the configuration key for a data entity is enabled, and the underlying tables configuration keys are enabled, the data management framework will check the configuration key on the fields in the tables. If the configuration key for a field is disabled, that field will not be available in the data entity for functional use even if the corresponding data entity field has the configuration key enabled.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Enabled</td>
<td>Enabled</td>
<td>Disabled</td>
<td>If the configuration key is enabled at all other levels, but the entity field configuration key is not enabled, then the field will not be available for use in the data entity.</td>
</tr>
</tbody>
</table>

**NOTE**

If an entity has another entity as a data source, then the above semantics are applied in a recursive manner.

**Entity list refresh**

When the entity list is refreshed, the data management framework builds the configuration key metadata for runtime use. This metadata is built using the logic described above. We strongly recommend that you wait for the entity list refresh to complete before using jobs and entities in the data management framework. If you don’t wait, the configuration key metadata may not be up to date and could result in unexpected outcomes. When the entity list is being refreshed, the following message is shown in the entity list page.
Data entity list page

The data entity list page in the Data management workspace shows the configuration key settings for the entities. Start from this page to understand the impact of configuration keys on the data entity.

This information is shown using the metadata that is built during entity refresh. The configuration key column shows the name of the configuration key that is associated with the data entity. If this column is blank it means that there is no configuration key associated with the data entity. The configuration key status column shows the state of the configuration key. If it has a checkmark, it means the key is enabled. If it is blank, it means either the key is disabled or there is no key associated.

Target fields

The next step is to drill into the data entity to view the impact of configuration keys on tables and fields. The target fields form for a data entity shows the configuration key and the key status information for the related tables and fields in the data entity. If the data entity itself has its configuration key disabled, a warning message is shown informing that the tables and fields in the target fields form for this entity will not be available, regardless of their configuration key status.

Child entities

Certain entities have other entities as data sources, or are composite data entities: configuration key information for these entities is shown in the Child entities form. Use this form in the similar way to the entities list page.
Run time validations for configuration keys

Using the configuration key metadata built during entity refresh list, run time validations are performed in the following use cases.

- When a data entity is added to a job.
- When user clicks Validate on the entity list.
- When the user loads a data package into a data project.
- When the user loads a template into a data project.
- When an existing data project is loaded.
- When a template is loaded into a data project.
- Before the export/import job is executed (batch, non-batch, recurring, OData).
- When the user generates mapping.
- When the user maps fields in the mapping UI.
- When the user adds only ‘importable fields’.

Managing configuration key changes

Anytime that you update configuration keys at the entity, table, or field level, the entity list in the data management framework must be refreshed. This process ensures that the framework picks up the latest configuration key settings. Until the entity list is refreshed, the following warning will be shown in the entity list page. The updated configuration key changes will take effect immediately after the entity list is refreshed. We recommend that you validate existing data projects and jobs to make sure that they function as expected after the configuration keys changes are put in effect.
Configuration data projects are used to manage the movement of company configuration data between instances of your application. They are intended to support the following scenarios:

- **Export of configurations** – Create configurations of entities, and use the data management framework to export the configurations to a package.
- **Import of configurations** – Upload a configuration package, and use the data management framework to import the package.

Configuration data packages are created by using data import and export projects in the Data management workspace. The **Data import** and **Data export** pages let you add and remove the entities that you require in order to manage the movement of company and shared data. After you create the list of entities in your configuration, you can export or import the configuration by using the data management framework to create a package. You can export packages locally and then move them to another instance for import.

Configuration data templates are predefined lists of entities for each module area that can be used in a data project. You can create, view, and modify these templates by using the **Template** page in the Data management workspace.

**IMPORTANT**

Default configuration templates were delivered in Microsoft Dynamics 365 for Finance and Operations, Enterprise edition July 2017 update. The Configuration data project feature is available in Microsoft Dynamics 365 for Operations platform update 7. You can create and use your own templates in the current product release.

Process for working with configuration data projects

We recommend that you follow this process when you start to use configuration data projects.

1. Set up your system by getting the new data configuration user interface (UI) and setting default file extensions.
2. Set up configuration templates for both export and import.
3. Create and run a configuration data project for export.
4. Create and run a configuration data project for import.

The rest of this topic describes the Data management workspace and explains how to set up your system for data configuration projects.

If you're ready to set up a configuration template, see Configuration data templates. If you're ready to create and run a configuration data project, see Copy configuration data between companies or legal entities overview.

Data management workspace overview

The Data management workspace provides access to important tasks for data management. It also provides information about projects and project execution tasks.

After you've created a configuration data project, it appears in the Data projects grid in the Data management workspace. For each project, the type of configuration (import or export) and the project category (Project, Configuration, Integration, or Other) are shown. Use the selections to the left of the grid
Set up your system to manage configuration data projects

Before you begin, make sure that you have access to the new data configuration UI, and that you’ve set up the default data sources.

Get the new UI

We have updated the Data management workspace, and the Template, Data export, and Data import pages, so that they support configuration management. However, the updated pages are available only in Enhanced view. Standard view hasn’t been updated from previous releases.

Change to the new UI for a single user

The system’s default view settings can be changed to save settings for each user per each legal entity. To switch the view for a single user, the user must select the Enhanced view button on each page in the Data management workspace.

Change to the new UI for all users in all legal entities

1. In the Data management workspace, select the Framework parameters tile.
2. Change the View default setting from Standard to Enhanced for all users and legal entities.

Set file extensions for default data sources

The Template and Import/export pages let you add entities by selecting a file that was created by using the data management framework. We have added the following default file name extensions for the various types of data sources:

- .xlsx for Microsoft Excel data sources
- .zip for package data sources
- .xml for XML data sources
- .csv for comma-separated values (CSV) data sources
- .txt for delimited data sources

If you want to use other file name extensions, you must update your data sources so that they have a default file name extension that will be used as the target data format.

1. In the Data management workspace, select the Configure data sources tile, and then select Data sources.
2. Add a default file name extension to the appropriate data source.
There are two options for copying configuration data in Finance and Operations:

- To move data between instances, you must first export it from one company and then import it to another company.
- To move data from one legal entity to another legal entity in the same instance, you can use the **Copy into legal entity** feature.

### Export a configuration

The **Data management** workspace is your hub for managing configuration data projects and exporting data packages. To build a configuration, you must define a data project and export the information that is represented by entities.

To create an export configuration data project, follow these steps:

1. Open the **Data management** workspace. If you're in Standard view, select **Enhanced view**.
2. Select the **Export** tile.
3. Select **New** to create an export configuration data project, and enter an ID and name for the configuration.
4. Set the operation type for the data project to **Export**, and set the project category to **Configuration**.
5. Add the entities that represent the information that you want to export. You can add entities by using several methods:
   - **Add one entity** – Enter the first part of the name of the entity until it appears in the lookup.
   - **Add multiple entities** – Enter any part of the entity name, use the lookup for the module, enter any part of the tag name, or use the lookup for the entity category to show a list of entities. Press Tab to move focus away from the **Lookup** field and activate the filter. In the grid, select the entities to add.
   - **Add a file** – Browse to a file that contains a name that matches the name of an entity and a file name extension that matches the file name extension that is in your data sources.
   - **Add a template** – Select from a list of templates that you've loaded in your instance.
6. Select a target data format. The system stores the last data format that you selected. Alternatively, if you select a file, the system automatically sets the data format to the data source that matches the file name extension.

**NOTE**

Composite entities require XML format.

7. Select **Add**. If you load a template, and the project already includes an entity that matches an entity in the template, the entity in the project will be replaced by the entity in the template. Some templates are very large, and they might take a few seconds to be loaded.
8. Select **Remove entity** to remove one or more selected entities.
9. When you’ve completed the configuration, select Export to start the export. You can monitor the results on the **Execution summary** page that appears.

Before you export a configuration, you might want to use some additional features that can help control the export process:

- To organize the list, use the **Sort by** button to reorder the entities by unit, level, and sequence.
- To change the execution sequence of any of the entities, you can manually edit the unit, level, or sequence. Alternatively, you can use the **Resequence** button to update any entities that you’ve selected. The **Resequence** button appears only if you select more than one entity. You can change the unit, level, and sequence individually. Alternatively, you can enable multiple changes and make them all at the same time. If you want the unit, level, or sequence to remain unchanged when you change multiple parts of the sequence, set the increment to 0 (zero).
- To add filters to the entity, use the **Filter** button. If you add a filter, the **Filter** button changes to an **Edit** button. The data will be filtered before it’s exported. If you’ve added a template, and the template includes filters, those filters will be added to your project. However, you can modify or remove them as you require.
- If you must change the entity mappings, use the **View map** button. If you’ve added a template, and the template includes mapping changes, those changes will be applied to your project. However, you can modify them as you require.
- To temporarily prevent the entity from being used when you export a data project, use the check box in the **Disable** column.
- To open the contents of the grid in a Microsoft Excel workbook, use the **Open in Excel** button. Modify the entities as you require, and then use the **Publish** button to upload the changes back.

When the export is completed, complete the following tasks:

- Use the **Download** button on the **Export** page to download the configuration settings.
- Use the **Download package** button in the **Data management** workspace or on the **Execution summary** page to download the configuration settings and the data that was exported.

### Setup considerations for some entities that are used to export configurations

Currently, several entities require additional steps when you build configurations. Follow these recommendations as you build your configurations.

**IMPORTANT**

This list will be updated as the Copy configuration feature is improved.

### Using special-purpose entities

The following entities require special handling when they are used in configurations.

<table>
<thead>
<tr>
<th>AREA</th>
<th>ENTITY</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>System setup</td>
<td>Global address book</td>
<td>The entity no longer exports the records that are created automatically when a company is created. The import no longer accepts those records.</td>
</tr>
<tr>
<td>AREA</td>
<td>ENTITY</td>
<td>ACTION</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GL Shared</td>
<td>Account structures active group</td>
<td>This composite entity will export and import only the active account structures. If you use any other account structure entities, the status of the active account structures will be changed to <strong>Draft</strong>, and you must activate them before they can be used.</td>
</tr>
<tr>
<td></td>
<td>Advanced rule structures active group</td>
<td>This composite entity is used in combination with the account structures active group entity. It will export and import only the active advanced rule structures. If you use any other advanced rule structure entities, the status of the advanced rule structures will be changed to <strong>Draft</strong>, and you must activate them before they can be used.</td>
</tr>
<tr>
<td></td>
<td>Financial dimension values</td>
<td>All dimension values will be exported, even values that are based on system-defined entities such as projects or customers. Remove the system-defined values before you import them. If you leave the system-defined values in the package, they won’t be imported. However, they will be filled as you import the data that backs the system-defined dimension.</td>
</tr>
<tr>
<td>Workflow</td>
<td>Workflow version</td>
<td>Change the owner for every record in the package data to <strong>Admin</strong>, unless the users in the workflow are already imported.</td>
</tr>
<tr>
<td></td>
<td>Workflow expression</td>
<td>Some workflow expressions might be too long for an Excel cell. Use XML instead of Excel as the export format.</td>
</tr>
<tr>
<td>Tax</td>
<td>Sales tax parameters</td>
<td>The default value for the marginal base calculations method is <strong>Total</strong> for sales tax parameters. The Ledger Parameters entity doesn’t set that value. However, the marginal base that some tax codes use, <strong>Line</strong>, will fail validation. A new entity that is named the Sales tax parameters preset entity was created so that you can import the marginal base calculation method first. You can then import tax codes.</td>
</tr>
<tr>
<td>AREA</td>
<td>ENTITY</td>
<td>ACTION</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Accounts receivable</td>
<td>Customers</td>
<td>The Customers entity was designed to be used for OData scenarios. For configurations, use the Customer definitions entity and the Customer details entity. The Customer definitions entities let you import the basic information about a customer. In this way, you enable entities that require that a customer have that information earlier in the import process. The Customer details entity contains additional information about a customer that you can add after parameters and reference data have been set up.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inventory management</td>
<td>Warehouse locations</td>
<td>Some warehouse locations require a location profile ID. Location profile IDs require a location format. Currently, the location format information must be manually added before the warehouse location. The entities for the location format and location profile were added in version 7.2.3, (App update 3 of the July 2017 release).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product information management</td>
<td>Products</td>
<td>The Products and Released Products entities should be used for configurations. The Product master and Released product master entities should be used for OData scenarios.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Product document attachments</td>
<td>For attachments to product documents released product documents, you must never skip staging, because additional steps are performed in the staging environment. You must use a data package for export and import, because the export file must be accompanied by a resources folder that contains the attachments. The entities support images, documents, notes, and links. When you export, you will see an image file that has a name that resembles a globally unique identifier (GUID). This file is a valid data package that is required in order to complete the import.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Product attribute values</td>
<td>Product attribute values are assigned only when a user opens the Attribute values page from the Products details page. Currently, you can't import the values unless this step was performed in the golden build.</td>
</tr>
</tbody>
</table>
## Procurement

### Vendor catalog

See the "Vendor catalogs import" section of [Vendor catalogs in Dynamics AX](https://www.microsoft.com) on the Supply Chain Management blog.

## Project

### Shared category

The Shared category entity now exposes the following fields: `CATEGORYID`, `CATEGORYNAME`, `EXPENSETYPE`, `USINEXPENSE`, `USINPRODUCTION`, and `USEINPROJECT`. If you change the value of the `USEINEXPENSE` field to yes, the `EXPENSETYPE` should be set to one of the valid values that are available in the `Expense type` field on the Shared category page.

### Remove the mapping and apply filters for specific entity fields

In a golden build, customer-specific fields might not be set up. To help guarantee that the import works, you should unmap those fields. For example, workers are stored in many tables, but they might not be set up in a golden build. Filters might also be required for some fields in an entity.

The following entities might have to be unmapped or filtered.

<table>
<thead>
<tr>
<th>AREA</th>
<th>ENTITY</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>System setup</td>
<td>Operating unit</td>
<td>Unmap Manager personnel number unless workers have been imported.</td>
</tr>
<tr>
<td></td>
<td>User information</td>
<td>Apply a filter where ID isn't equal to Admin. Unmap Person name, and use the User to person relationship entity to map system users to directory users.</td>
</tr>
<tr>
<td>Accounts payable</td>
<td>Vendors</td>
<td>Unmap Purchase site (DefaultPurchaseSite) and Warehouse (DefaultProcurementWarehouseID) unless they are set up. Unmap the vendor bank account ID. The Vendor bank account entity will set up the link to the bank account when it's imported.</td>
</tr>
<tr>
<td>Accounts receivable</td>
<td>Customer details</td>
<td>Unmap Employee responsible number unless workers have been imported. Unmap Collections contact person (CollectionsContactPersonID) unless workers and their contact information have been imported. Unmap the site (SiteID) and warehouse (WarehouseID) unless they have already been imported.</td>
</tr>
<tr>
<td>Inventory management</td>
<td>Warehouse current postal address</td>
<td>Unmap Picking store area and Input store area unless Commerce information has been imported.</td>
</tr>
<tr>
<td>AREA</td>
<td>ENTITY</td>
<td>ACTION</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Product information management</td>
<td>Products</td>
<td>Unmap NMFCCode and STCCCode unless you are adding the transportation management template to your data project.</td>
</tr>
<tr>
<td></td>
<td>Released products</td>
<td>Unmap the project category, default product color, default configuration, default product size, and default product style. This entity is self-referencing and hasn't yet been updated to load these fields in a single pass.</td>
</tr>
<tr>
<td></td>
<td>Period template</td>
<td>The Period template entity is a shared entity. Although it can be filtered by legal entity, the Period template lines entity doesn't have a Legal entity field. To import a single legal entity, you can filter the period template. However, you must currently remove the period template lines that aren't related to that legal entity.</td>
</tr>
<tr>
<td></td>
<td>Item coverage group</td>
<td>Unmap Period template ID unless it has already been added manually.</td>
</tr>
<tr>
<td>Procurement</td>
<td>Vendors</td>
<td>Unmap Purchase site (DefaultPurchaseSite) and Warehouse (DefaultProcurementWarehouseID) unless they are set up. Unmap the vendor bank account ID. The Vendor bank account entity will set up the link to the bank account when it's imported.</td>
</tr>
<tr>
<td>Sales and marketing</td>
<td>Leads</td>
<td>Unmap LeadOpeningPersonnelNumber, LeadClosingPersonnelNumber, and LeadResponsiblePersonnelNumber unless workers have been imported.</td>
</tr>
<tr>
<td>Sales type document entry policies</td>
<td></td>
<td>Unmap IsAtpGenrallyIncludingPlannedOrders. The default for Master planning was changed to be Yes for Disable all planning processes when a legal entity is created.</td>
</tr>
<tr>
<td>Project management</td>
<td>Projects</td>
<td>Unmap WorkerArchitectPersonelNumber, WorkerRespFinancialPersonelNumber, WorkerRespSalesPersonelNumber, and WorkerRespSalesPersonelNumber unless workers have been imported.</td>
</tr>
</tbody>
</table>
### Golden builds that have multiple legal entities

The templates can be used to export data from any company in your golden build. When you export both shared data and company data, the templates first export the shared data for all legal entities. They then export the data for the legal entity that you're currently using. You can then switch companies and export the company data for additional legal entities by using projects that don't include the shared entities.

If you're exporting from a golden build that contains multiple legal entities, but you want to import the data from only one of those legal entities, you must apply a filter on the legal entity fields, so that only the data that you require for that legal entity is exported. This filter must remove all data for all legal entities except the legal entity that you want. In some cases, you must complete additional steps to clean up the exported data.

Most of the changes that are listed in the following table occur in the **System setup** and **General ledger** areas. If you export a golden build that uses a single legal entity, you should not require these filters.

The following entities require filters or special handling when you export the data.

<table>
<thead>
<tr>
<th>AREA</th>
<th>ENTITY</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>System setup</td>
<td>Legal entities</td>
<td>Apply a filter to Company.</td>
</tr>
<tr>
<td>AREA</td>
<td>ENTITY</td>
<td>ACTION</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Number sequence code          |                                                                        | The number sequence codes can be shared, or they can be specific to a legal entity. To import all number sequences, you must have the legal entities setup for the number sequences that are stored for a specific legal entity. If you want only shared number sequences, apply a filter to remove the number sequences that are specific to a legal entity. If you want the number sequences only for a specific legal entity, apply a filter to Company. Number sequences can also have scope. You must delete number sequences when the following conditions are met:  
  - The SCOPETYPE value is DataArea, and the SCOPEVALUE value either isn't equal to the legal entity or is blank.  
  - The SCOPETYPE value is LegalEntity, and the SCOPEVALUE value isn't equal to the legal entity.  
  - The SCOPETYPE value is DataAreaFiscalCalendar, and the SCOPEVALUE value isn't equal to the legal entity.
| Number sequence references    |                                                                        | Number sequence references can also have scope. You must delete the number sequence references when the following conditions are met:  
  - The SCOPETYPE value is equal to DataArea, DataAreaFiscalCalendar, or LegalEntity.  
  - The SCOPEVALUE value either isn't equal to the legal entity that you want in the data or is blank. |
<p>| Organization hierarchies      |                                                                        | There is no legal entity filter. Manually remove any references to the other legal entities that you aren't importing. For example, if you've set up centralized payments, but you're loading only one legal entity, you must not import the Centralized payments hierarchy. |</p>
<table>
<thead>
<tr>
<th>AREA</th>
<th>ENTITY</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global address book</td>
<td>Multiple entities</td>
<td>The global address book contains data for all legal entities. All legal entities will be created when you import the data, unless you remove the data for the legal entities that you don't want to load. To help guarantee that other legal entities aren't created, delete all records for those other legal entities. Alternatively, apply a filter to the Legal entity data area (&lt; &gt; blank). You must also remove global address book records that are used in the other legal entities.</td>
</tr>
<tr>
<td>Party postal addresses</td>
<td></td>
<td>Address records for legal entities other than the legal entity that you're importing must be manually deleted before import.</td>
</tr>
<tr>
<td>Party contacts</td>
<td></td>
<td>Contact records for legal entities other than the legal entity that you're importing must be manually deleted before import.</td>
</tr>
<tr>
<td>Workflow</td>
<td></td>
<td>Apply a filter on DataAreaID. Many of the workflow entities can't be filtered for legal entity. Therefore, import failures might occur if you don't remove the data in all workflow entities that are related to legal entities that you don't want. We recommend that you manually set up workflows for this scenario.</td>
</tr>
<tr>
<td>General ledger</td>
<td>Ledger</td>
<td>Apply a filter to Company.</td>
</tr>
<tr>
<td></td>
<td>Ledger fiscal calendar year</td>
<td>Apply a filter to Ledger name.</td>
</tr>
<tr>
<td></td>
<td>Ledger fiscal calendar period</td>
<td>Apply a filter to Ledger name.</td>
</tr>
<tr>
<td></td>
<td>Main account legal entity overrides</td>
<td>Apply a filter to Company.</td>
</tr>
<tr>
<td></td>
<td>Financial dimension value legal entity</td>
<td>Apply a filter to Legal entity.</td>
</tr>
<tr>
<td></td>
<td>Financial dimension values</td>
<td>Apply a filter to Legal entity. However, you can't filter out system-defined dimension values. Therefore, you must delete them. Those system-defined dimension values will be restored when you import data for the tables that back the system-defined dimensions.</td>
</tr>
<tr>
<td></td>
<td>Journal names</td>
<td>Apply a filter to Voucher series company ID.</td>
</tr>
<tr>
<td></td>
<td>Ledger allocation basis source</td>
<td>Apply a filter to Legal entity.</td>
</tr>
<tr>
<td>AREA</td>
<td>ENTITY</td>
<td>ACTION</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Ledger allocation rule destination</td>
<td>Apply a filter to Company.</td>
<td></td>
</tr>
<tr>
<td>Accounts receivable</td>
<td>Customer write-off reason code</td>
<td>Apply a filter to Company.</td>
</tr>
<tr>
<td>Budget</td>
<td>Budget control configuration</td>
<td>Apply a filter to Legal entity.</td>
</tr>
<tr>
<td></td>
<td>Budget control configuration activation</td>
<td>Apply a filter to Legal entity.</td>
</tr>
<tr>
<td></td>
<td>Budget control cycle model</td>
<td>Apply a filter to Legal entity.</td>
</tr>
<tr>
<td></td>
<td>Budget control dimension attribute</td>
<td>Apply a filter to Legal entity.</td>
</tr>
<tr>
<td></td>
<td>Budget control documents and journals</td>
<td>Apply a filter to Legal entity.</td>
</tr>
<tr>
<td></td>
<td>Budget control group</td>
<td>Apply a filter to Legal entity.</td>
</tr>
<tr>
<td></td>
<td>Budget control group criteria</td>
<td>Apply a filter to Legal entity.</td>
</tr>
<tr>
<td></td>
<td>Budget control message level</td>
<td>Apply a filter to Legal entity.</td>
</tr>
<tr>
<td></td>
<td>Budget control over budget permissions</td>
<td>Apply a filter to Legal entity.</td>
</tr>
<tr>
<td></td>
<td>Budget control rule</td>
<td>Apply a filter to Legal entity.</td>
</tr>
<tr>
<td></td>
<td>Budget control rule criteria</td>
<td>Apply a filter to Legal entity.</td>
</tr>
<tr>
<td></td>
<td>Budget cost elements</td>
<td>Apply a filter to Cost element data area ID.</td>
</tr>
<tr>
<td></td>
<td>Budget dimensions</td>
<td>Apply a filter to Legal entity.</td>
</tr>
<tr>
<td></td>
<td>Budget plan allocation schedule</td>
<td>Apply a filter to Ledger.</td>
</tr>
<tr>
<td></td>
<td>Budget plan process</td>
<td>Apply a filter to Ledger.</td>
</tr>
<tr>
<td></td>
<td>Budget plan stage rule</td>
<td>If you applied a filter to Budget plan process, errors might occur when you import stage rules. Because the entity doesn't currently contain a ledger name that can be filtered, it will contain all companies.</td>
</tr>
<tr>
<td></td>
<td>Budget plan priority constraint</td>
<td>You will experience the same issue that is described for the Budget plan stage rule entity.</td>
</tr>
<tr>
<td></td>
<td>Budget plan process administration</td>
<td>You will experience the same issue that is described for the Budget plan stage rule entity.</td>
</tr>
</tbody>
</table>
### Changing the legal entity value before import

If you want to change the legal entity ID to another value, the value of all fields that resemble the fields that were listed earlier must be changed to the value of the new legal entity. For example, for Legal entities, change Company from the exported value to a new value in the exported file.

The legal entity ID is stored in many places. Therefore, it can be difficult to make this change, and you might cause errors if you try.
Selecting a single legal entity for export

To export a single legal entity, you can create a Copy into legal entity data project and specify the legal entity to copy as the source legal entity. When you add entities or load templates, that type of project will automatically add legal entity filters. You can then download the package or create a template from it on the Templates page. The package or template can then be added to an export project and used to export the legal entity.

Import a configuration

The Data management workspace is also your hub for importing configuration data projects. You can build a configuration project from an existing data project that you exported. Alternatively, you can build a configuration project from individual files that contain data that is formatted correctly for import.

To import a configuration, follow these steps.

1. Open the Data management workspace. If you're in Standard view, select Enhanced view.
2. Select the Import tile.
3. Select New to create a configuration data project, and enter an ID and name for the configuration.
4. Set the operation type for the data project to Import, and set the project category to Configuration.
5. Add the entities that represent the information that you want to copy. You can add entities by using several methods:
   - Add one entity – Enter the first part of the name of the entity until it appears in the lookup.
   - Add multiple entities – Enter any part of the entity name, use the lookup for the module, enter any part of the tag name, or use the lookup for the entity category to show a list of entities. Press Tab to move focus away from the Lookup field and activate the filter. In the grid, select the entities to add.
   - Add a file – Browse to a file that contains a name that matches the name of an entity and a file name extension that matches the file name extension that is in your data sources, and the source data format will be set automatically. If you haven’t set up the default file name extensions, you must select a source data format before you select the file.
   - Add a template – Select from a list of templates that you’ve loaded in your instance.

When you load a package, the Import page first reads the list of entities from the package. A progress indicator shows how much of the package has been read. After the list of entities is read, the Import page starts to load the data in the package. This process can take some time.

6. Select Remove entity to remove any selected entities, as required.
7. After you’ve completed the configuration, select Import to start the import. You can monitor the results on the Execution details page that appears.

Before you import a configuration, you might want to use some additional features that can help control the import process:

- To organize the list, use the Sort by button to reorder the entities by unit, level, or sequence.
- To change the execution sequence of any of the entities, you can manually edit the unit, level, or sequence. Alternatively, you can use the Resequence button to update any entities that you’ve selected.
- If you must change the entity mappings, use the View map button.
To temporarily prevent the entity from being used when you export a data project, use the check box in the **Disable** column.

**Copy into a legal entity**

The **Data management** workspace is also your hub for copying configuration information from one legal entity to another. The process resembles an export and import that occur in one step. As in an import, if information that exists in the source legal entity doesn’t exist in the destination legal entity, the copy process adds it. If information already exists in the destination legal entity, the copy process updates it.

To copy a configuration from one legal entity to another legal entity in the same instance, follow these steps.

1. Open the **Data management** workspace. If you’re in Standard view, select **Enhanced view**.
2. Select the **Copy into legal entity** tile.
3. Select **New** to create a configuration data project, and enter an ID and name for the configuration.
4. Set the operation type for the data project to **Copy into legal entity**, and set the project category to **Configuration**.
5. Select the legal entity that should be the source of the data to copy. By default, the legal entity that you’re currently using is selected.
6. On the **Legal entities** FastTab, you can select existing legal entities as a destination, or you can create new legal entities:
   - **Select** – Select one or more legal entities in the list, and then select **Add selected**. The legal entities are added to the list of destination legal entities.
   - **Create** – Enter the legal entity ID, the legal entity name, and the region that the legal entity belongs in. Then select **Create legal entity**. The legal entity is created and added to the list of destination legal entities.

**NOTE**

The functionality for creating destination legal entities is available in Finance and Operations 7.2.3.

7. After you’ve added the destination legal entities, select **Yes** if the number sequences should be copied. The entities that are required in order to copy the number sequence codes and number sequence references will be added to the project. The execution unit, level, and sequence number for these entities are set to the numbers in the default System and Shared templates. If you aren’t using the default templates, adjust the entity sequences so that they are first in the list.
8. If you selected **Yes** for number sequences, select **Yes** or **No** to specify whether those number sequences should be reset to the smallest value.
9. Add the entities that represent the information that you want to copy. You can add entities by using several methods:
   - **Add one entity** – Enter the first part of the name of the entity until it appears in the lookup.
   - **Add multiple entities** – Enter any part of the entity name, use the lookup for the module, enter any part of the tag name, or use the lookup for the entity category to show a list of entities. Press Tab to move focus away from the **Lookup** field and activate the filter. In the grid, select the entities to add.
   - **Add a file** – Browse to a file that contains a name that matches the name of an entity and a file name extension that matches the file name extension that is in your data sources.
   - **Add a template** – Select from a list of templates that you’ve loaded in your instance.
To help guarantee that the correct order is maintained, we recommend that you use the default templates. Then add and remove entities to match the data that you want to copy. You can remove the entities that you don’t want to copy.

**NOTE**
- If an entity has a field that represents the legal entity, a filter will be applied to that entity, so that only the data for the source legal entity is included. The value for that field will be changed to the destination legal entity.
- Document, transaction, and composite entities aren't available when you copy to a legal entity.

10. Select **Remove entity** to remove any selected entities, as required.

11. After you’ve completed the configuration, select **Copy into legal entity** to start the import. The copy process will export the data from the source legal entity into the destination legal entity. Each destination legal entity will have its own import data project. You can monitor the results on the **Execution summary** page that appears. All import projects that are related to the Copy into legal entity project will appear in a list on the left of the page.

Any errors that occur are shown on the **Execution summary** page, just as they are for an import project. You can edit the errors in the staging tables and resubmit the values for each data project.

**Special considerations when you copy into a legal entity**

When you copy into a legal entity, you have the same validation that occurs when you import a file. It’s important that you test your copy in a test environment, so that you can identify any dependencies that will cause failures. If dependent information isn’t included in your list of entities to copy, the entity will show errors when it tries to copy into the legal entity. For example, if a customer has a default site or warehouse, you must use one of these approaches:

- Import the sites and warehouses as part of the copy.
- Manually load the sites and warehouses before you copy the legal entity.
- Unmap the site and warehouse fields before you copy the information.

You might also experience import errors if you copy from one region to another region. For example, you can have 1099 fields in a legal entity in the US region. However, if you try to import those values into a legal entity in a German region, you will see errors on the import. If a template that you load has entities that are incorrect for a region, you will receive an error message that states that the incorrect entity wasn’t loaded. However, the rest of the template will continue to be loaded. You should copy only information that is appropriate for the destination region.

The following entities require special handling when they are used to copy into a legal entity.
<table>
<thead>
<tr>
<th>AREA</th>
<th>ENTITY</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>System setup</td>
<td>Number sequences</td>
<td>If you use a number sequence for vendors and customers on the parameters forms and then you copy customers and vendors, you need to make sure that the “Allow user to change to a lower number” settings on the number sequences to Yes. The “No” settings will cause the import to reject vendor and customer numbers since they were created before using numbers lower than the next available number sequence. If you use the Reset to smallest feature, you need to change the “Allow user to change to a higher number” since the vendor and customer numbers are higher than the next available number sequence.</td>
</tr>
<tr>
<td>System setup</td>
<td>Workflow</td>
<td>Workflow requires additional changes before it can be copied. Workflow copies are not supported at this time.</td>
</tr>
<tr>
<td>Accounts payable</td>
<td>Vendors</td>
<td>Vendors have many settings that are dependent on the values that come from other entities. For example, if you update the matching settings to require three-way matching but you have vendors set for two-way matching, the vendor will fail validation. If you use auto sequencing for new vendor, you will need to unmapped vendor account before you do the copy into legal entity. In addition, if an auto-sequenced vendor has an invoice account, that invoice account is not transformed and may fail. You may see similar issues in other areas such as vendors in posting accounts and approved vendor list by products.</td>
</tr>
<tr>
<td>Accounts receivable</td>
<td>Customers</td>
<td>Customers have many settings that are dependent on the values that come from other entities. For example, if you have a default warehouse and site for a customer, you must add sites and warehouses first or the customer will fail validation. The collections contact will also fail if the contact is not available in the new company. If you use auto sequencing for new customers, you will need to unmapped the customer account before you do the copy into legal entity. In addition, if an auto-sequenced customer has an invoice account, that invoice account is not transformed and may fail. You may see similar issues in other areas such as customers in posting accounts.</td>
</tr>
<tr>
<td>AREA</td>
<td>ENTITY</td>
<td>ACTION</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Budget</td>
<td>Budget cost elements</td>
<td>There is an issue when importing budget cost elements when using an annual amount and there are budget cost elements in the Earnings basis tab. The issue will be addressed in a future release.</td>
</tr>
<tr>
<td>Fixed assets</td>
<td>Fixed assets depreciation profile manual schedule</td>
<td>The fixed asset depreciation profile must be processed first. Change the sequence on your data project for fixed asset depreciation profile to 15 (instead of 10). We will update the default templates in the monthly application release 4 to this value.</td>
</tr>
<tr>
<td>General ledger</td>
<td>Intercompany accounting</td>
<td>The copy into legal entity feature does not support intercompany accounting at this time. The issue will be addressed in a future release.</td>
</tr>
<tr>
<td>General ledger</td>
<td>Journal names</td>
<td>Only the journal names for the source legal entity will be copied to the destination companies. If you select the lookup in the journal names form, you will only see the number sequences that are available for that legal entity. However, if you choose to enter a number sequence manually that is not on that list, it will not be included in the copy process.</td>
</tr>
<tr>
<td>General ledger</td>
<td>Ledger allocation rules destination</td>
<td>If you are running a version earlier than version 7.3, if you have cross company allocation rules, you will not see the destination rules for legal entities that do not match the source legal entity. The copy into legal entity feature will only copy the destination records when the destination is equal to the source. The issue has been resolved in version 7.3.</td>
</tr>
<tr>
<td>General ledger</td>
<td>Ledger fiscal calendar year/period</td>
<td>The process is currently exporting all of the legal entities instead of just the source legal entity. The copy process works correctly. The issue has been resolved in version 7.3.</td>
</tr>
<tr>
<td>AREA</td>
<td>ENTITY</td>
<td>ACTION</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>General ledger</td>
<td>Ledger parameters</td>
<td>If you check for continuous number sequences but you have number sequences that are used on journal names and are not continuous, the imports will fail. You should temporarily turn off that setting in the ledger parameters. In addition, the ledger parameters must be processed first. Change the sequence on your data project for ledger parameters to 15 (instead of 40). We will update the default templates in the monthly application release 3 to this value.</td>
</tr>
<tr>
<td>Inventory management</td>
<td>Inventory dimension parameters</td>
<td>There is an issue where an error on import is shown but the correct number of parameters are imported. The issue will be addressed in a future release.</td>
</tr>
<tr>
<td>Inventory management</td>
<td>Warehouse locations</td>
<td>Some warehouse locations require a location profile ID. Location profile IDs require a location format. Currently, the location format information must be manually added before the warehouse location. The entities for the location format and location profile were added in version 7.2.3, (App update 3 of the July 2017 release).</td>
</tr>
<tr>
<td>Master planning</td>
<td>Intercompany master plan associations</td>
<td>The copy into legal entity feature does not support intercompany master plan associations at this time. The issue will be addressed in a future release.</td>
</tr>
<tr>
<td>Retail</td>
<td>POS registers</td>
<td>This entity is global and cannot be copied to another legal entity.</td>
</tr>
<tr>
<td>Retail</td>
<td>Retail channel</td>
<td>This entity is global and cannot be copied to another legal entity.</td>
</tr>
<tr>
<td>Retail</td>
<td>Retail store address book</td>
<td>This entity has a dependency on Retail channel so it cannot be used.</td>
</tr>
<tr>
<td>Sales and marketing</td>
<td>Intercompany trading partnerships</td>
<td>The copy into legal entity feature does not support intercompany trading partnerships at this time. The issue will be addressed in a future release.</td>
</tr>
</tbody>
</table>

**Rules**

The Copy into legal entity feature supports rules, which let you modify data before it’s added to the import staging table. For example, rules are used to change the target legal entity to the destination legal entity and modify number sequences. You can extend rules.

Rule extensions require three changes:

1. Add the name of the rule to the extensible **DMFRulesType** enumeration (enum).
2. For each project definition group, insert the enum that you just created (for example, `DMFRulesType::NewRule`) into the DMFRules table. Use your source legal entity, your rule type, and the definition group name. If you require that more data be stored, such as various options for your rule, you can create your own table and extend the DMFRules table.

3. Create your own class to handle the data that the rule acts on. For the DMFRules framework to instantiate the rule, you must decorate the class with the `[DMFRulesBaseFactoryAttribute(DMFRulesType::NewRule)]` attribute.

The class must also extend the `DMFRulesBase` class. This extension will require an implementation of the `DMFRulesBase.runRule(Common_staging)` method. The _staging record will be the buffer of the staging record that the rule is applied to.

**Additional information about entities**

**Obsolete entities**

As the application is updated, the functionality of an entity might have to be updated. A new entity might be created that has a different name. In this case, the original entity will be marked as obsolete. You will no longer be able to add obsolete entities to a new data project or template.

If you load a data package that contains an obsolete entity, you will receive a warning about the existence of obsolete entities. However, you will still be able to import your data. You can find the obsolete entity by selecting the **Obsolete** column and filtering on **Yes**.

**Self-referencing entities**

Some entities represent tables that have references to themselves. For example, when you create a cash discount, you can refer to a related cash discount that creates a tiered discount calculation. To import data, you must sequence the data so that the cash discount that is referred to in the **Next cash discount** field is imported before the cash discount that uses that cash discount.

A new `DMFImportExportSequencer` class sequences the data in self-referencing entities and enables the data to be loaded in a single pass. In the Cash Discount entity (CashDiscountEntity), you can view the code that is required in order to update entities.

The class has been added to several self-referencing entities. It will be added to more entities as required. Here are some other examples of self-referencing entities:

- Customers
- Customer definitions
- Customer details
- Tax codes
- Budget control groups
- Projects
- Product categories
- Warehouses
- Budget plan workflow stage
- Sales units

**Adding entities in the appropriate country/region context**

When you add entities, the mappings are created in the context of the country or region of the company that is currently active. If there are any issues with the mappings, a red ‘X’ appears in the **View map** column. Select the red ‘X’ and repair the mappings as required.
NOTE

By default, the DAT company doesn't have a country/region context. Some entities, such as the entities that are used for transaction codes and 1099 fields, won't be mapped correctly if they are added to a data project for the DAT company, because a country/region context is expected.
Configuration data packages are available as process data packages from Microsoft Dynamics Lifecycle Services (LCS). These data packages can help improve the repeatability of implementations and accelerate the configuration.

Data packages contain configuration entity spreadsheets. These entity spreadsheets contain best practice data that you can use to create an initial golden build. The data entities in the data packages are also sequenced appropriately to help guarantee a successful single-click import of the data.

The entity spreadsheets include three types of data:

- **Business data** – The spreadsheet contains standard business data for a mid-sized trade or retail company. This data combines best practices and business standards that can be used as a starting point for your configuration.

- **Sample data** – The spreadsheet contains data that can be used as an example for business-specific data. This data can be imported and used as an example, but it must be changed for individual business practices.

- **No data** – The spreadsheet doesn’t contain any data. Several areas of the product are unique to each business and its business practices. These areas must be configured specifically for the organization. These spreadsheets should be reviewed and updated for the organization as appropriate.

For more information about the type of data that is included in each entity spreadsheet in the data packages see the Data packages section of this topic. You can modify individual spreadsheets before you import the data packages, or you can import the data packages as they have been supplied and then update your data in the system.

**Using configuration data packages**

You can access configuration data packages from LCS. You can either apply them to an LCS environment, or download them so that you can manually import them.

1. Open your LCS project, and open the Asset library.
2. In the list of asset types, select **Process data package**.
3. Click **Import**.
4. Select the configuration data package.
5. Click **Pick**.

At this point, you can use the **Consume** function to apply the process data package to an LCS environment.

You can also download the individual data package files from the Data package area. Use the Data management workspace to import the data packages from LCS. For more information about how to import and export configurations, see Copy configuration data between companies or legal entities overview.

**Special considerations**
**System setup**
The System data package must be imported before any other data package. By default, the System data package creates a new legal entity that is named **ST01**. The data packages for the module areas depend on this legal entity.

**General ledger**
A generic chart of accounts is included in the configuration data packages. When this data is used as it’s defined in the Main account entity spreadsheet, posting profiles across the system are filled with default posting data. If you change the main accounts that are used for the chart of accounts, you must also update the individual posting profiles and posting accounts for each area.

**Data packages: System**

<table>
<thead>
<tr>
<th>010 – System Setup</th>
<th>SPREADSHEET CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entity spreadsheet</strong></td>
<td><strong>Business data</strong></td>
</tr>
<tr>
<td>Address and contact information purpose</td>
<td>X</td>
</tr>
<tr>
<td>Address books</td>
<td></td>
</tr>
<tr>
<td>Address format lines</td>
<td>X</td>
</tr>
<tr>
<td>Address format</td>
<td>X</td>
</tr>
<tr>
<td>Address parameters</td>
<td>X</td>
</tr>
<tr>
<td>Calendar</td>
<td></td>
</tr>
<tr>
<td>Cities</td>
<td>X</td>
</tr>
<tr>
<td>Counties</td>
<td>X</td>
</tr>
<tr>
<td>CountryRegions</td>
<td>X</td>
</tr>
<tr>
<td>Currencies</td>
<td>X</td>
</tr>
<tr>
<td>Districts</td>
<td>X</td>
</tr>
<tr>
<td>Exchange rates</td>
<td></td>
</tr>
<tr>
<td>Global address book parameters</td>
<td>X</td>
</tr>
<tr>
<td>Global address book</td>
<td></td>
</tr>
<tr>
<td>Legal entities</td>
<td></td>
</tr>
<tr>
<td>Name affixes</td>
<td>X</td>
</tr>
<tr>
<td>Name sequences</td>
<td>X</td>
</tr>
<tr>
<td>----------------</td>
<td>---</td>
</tr>
<tr>
<td>Operating unit</td>
<td>X</td>
</tr>
<tr>
<td>Organization hierarchy purposes</td>
<td>X</td>
</tr>
<tr>
<td>Organization hierarchy type</td>
<td>X</td>
</tr>
<tr>
<td>Organization hierarchy</td>
<td>X</td>
</tr>
<tr>
<td>Party contacts</td>
<td>X</td>
</tr>
<tr>
<td>Party postal addresses</td>
<td>X</td>
</tr>
<tr>
<td>Party relationships</td>
<td>X</td>
</tr>
<tr>
<td>Postal codes</td>
<td>X</td>
</tr>
<tr>
<td>Relationship types</td>
<td>X</td>
</tr>
<tr>
<td>States</td>
<td>X</td>
</tr>
<tr>
<td>System parameters</td>
<td>X</td>
</tr>
<tr>
<td>Team types</td>
<td>X</td>
</tr>
<tr>
<td>Teams</td>
<td>X</td>
</tr>
<tr>
<td>Unit conversions</td>
<td>X</td>
</tr>
<tr>
<td>Unit translations</td>
<td>X</td>
</tr>
<tr>
<td>Units</td>
<td>X</td>
</tr>
<tr>
<td>User groups</td>
<td>X</td>
</tr>
<tr>
<td>User information</td>
<td>X</td>
</tr>
</tbody>
</table>

**Data packages: Financials**

**020 – GL Shared**

<table>
<thead>
<tr>
<th>Entity spreadsheet</th>
<th>Business data</th>
<th>Sample data</th>
<th>No data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account structure activation</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPREADSHEET CONTENT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Account structure allowed values</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Account structures</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced rule criteria</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced rule structure activation</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced rule structure allowed values</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced rule structures</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced rules</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chart of accounts</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consolidation groups and accounts</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dimension attribute activation</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Financial dimension format</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Financial dimension sets</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Financial dimension translations</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Financial dimension value translations</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Financial dimension values</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Financial dimensions</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fiscal calendar period</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fiscal calendar</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Main account categories</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Main account</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number sequence code</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number sequence group</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entity spreadsheet</td>
<td>Business data</td>
<td>Sample data</td>
<td>No data</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>Accounts for automatic transactions</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Balance control accounts</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Calendar</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Date intervals</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Default descriptions</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Dimension parameters</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Document types</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Financial dimension value legal entity overrides</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Financial reasons</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Journal descriptions</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Journal names</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Ledger allocation basis source</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Ledger allocation basis</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Ledger allocation rule destination</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Ledger allocation rule source</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Ledger allocation rule</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Ledger fiscal calendar period</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Ledger fiscal calendar year</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Ledger parameters</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>SPREADSHEET CONTENT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------</td>
<td>--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ledger</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Main account legal entity overrides</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organization email template message</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organization email template</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subledger journal transfer rule</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Workflow organization parameters</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Working times</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**100 – Bank**

<table>
<thead>
<tr>
<th>SPREADSHEET CONTENT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity spreadsheet</td>
<td>Business data</td>
</tr>
<tr>
<td>Bank accounts</td>
<td>X</td>
</tr>
<tr>
<td>Bank facility groups</td>
<td></td>
</tr>
<tr>
<td>Bank facility types</td>
<td></td>
</tr>
<tr>
<td>Bank groups</td>
<td></td>
</tr>
<tr>
<td>Bank parameters</td>
<td>X</td>
</tr>
<tr>
<td>Bank posting profiles</td>
<td></td>
</tr>
<tr>
<td>Bank statement import methods for general journal</td>
<td></td>
</tr>
<tr>
<td>Bank transaction groups</td>
<td>X</td>
</tr>
<tr>
<td>Bank transaction type</td>
<td>X</td>
</tr>
<tr>
<td>Check layout</td>
<td></td>
</tr>
<tr>
<td>Customer charge groups</td>
<td></td>
</tr>
<tr>
<td>Payment purpose codes</td>
<td></td>
</tr>
<tr>
<td>Reconciliation matching rule sets</td>
<td></td>
</tr>
<tr>
<td><strong>SPREADSHEET CONTENT</strong></td>
<td></td>
</tr>
<tr>
<td>-------------------------</td>
<td>--</td>
</tr>
<tr>
<td>Reconciliation matching rules</td>
<td>X</td>
</tr>
<tr>
<td>Transaction code mapping</td>
<td>X</td>
</tr>
</tbody>
</table>

### 120 – AP

<table>
<thead>
<tr>
<th><strong>SPREADSHEET CONTENT</strong></th>
<th><strong>Business data</strong></th>
<th><strong>Sample data</strong></th>
<th><strong>No data</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity spreadsheet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sample data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aging period definitions</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Business segments</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Business subsegments</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Buyer groups</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Cash discount</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Delivery charges groups</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Destination code</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Expedite codes</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Item charge groups</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Line discount vendor groups</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Line of business</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Modes of delivery</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Multiline discount vendor groups</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Payment calendar rule</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Payment calendar</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Payment day lines</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Payment schedule lines</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Payment schedule</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Price vendor groups</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td><strong>SPREADSHEET CONTENT</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Procurement charge codes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product categories</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product category hierarchies</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product category hierarchy roles</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product category hierarchy translations</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Purchase pools</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales tax groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terms of delivery</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terms of payment</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total discount vendor groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor bank accounts</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor charges group</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor exception groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor invoice form printing configurations</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor invoice policy rule types</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor ledger accounts</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor parameters</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor payment fee</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor payment format</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor payment method specification</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor payment method</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Vendor posting profile</strong></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Vendor price tolerance groups</strong></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Vendor reasons</strong></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Vendor, form parameters</strong></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Vendors</strong></td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 130 – Tax

<table>
<thead>
<tr>
<th><strong>Entity spreadsheet</strong></th>
<th><strong>Business data</strong></th>
<th><strong>Sample data</strong></th>
<th><strong>No data</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales tax authorities</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales tax code limits</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Sales tax code values</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Sales tax codes</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Sales tax exempt code</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Sales tax exempt numbers</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Sales tax group details</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Sales tax groups</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Sales tax item groups</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Sales tax ledger posting groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales tax parameters preset entity</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Sales tax parameters</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales tax period setup</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales tax reporting codes</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Transaction codes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Withholding tax code values</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Entity spreadsheet</td>
<td>Business data</td>
<td>Sample data</td>
<td>No data</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------</td>
<td>-------------</td>
<td>--------</td>
</tr>
<tr>
<td>Aging period definitions</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business segments</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Business subsegments</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Cash discount</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Collection letter form printing configurations</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Collection letter setup</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commission calculation</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Commission customer group</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Commission item group</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Commission sales group</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Customer account statement form printing configurations</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Customer bank accounts</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Customer charge groups</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Customer definitions</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Customer details</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Customer facing form printing configurations</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>SPREADSHEET CONTENT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer ledger accounts</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer parameters</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer payment fee</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer payment method</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer payment method</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer posting profiles</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer reasons</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer write-off reason</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customers</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delivery charges groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expedite codes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Free text invoice form</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interest codes with fees</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interest codes with ranges</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interest codes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interest note form printing</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item charge groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line discount customer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line of business</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modes of delivery</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payment calendar</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payment day lines</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPREADSHEET CONTENT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payment schedule lines</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payment schedule</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Price customer groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Printed form notes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales charge codes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales districts</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales order pools</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Statistics group</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terms of delivery</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terms of payment</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total discount customer groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Configuration data packages

Configuration data packages are available as process data packages from Microsoft Dynamics Lifecycle Services (LCS). These data packages can help improve the repeatability of implementations and accelerate the configuration.

Data packages contain configuration entity spreadsheets. These entity spreadsheets contain best practice data that you can use to create an initial golden build. The data entities in the data packages are also sequenced appropriately to help guarantee a successful single-click import of the data.

The entity spreadsheets include three types of data:

- **Business data** – The spreadsheet contains standard business data for a mid-sized trade or retail company. This data combines best practices and business standards that can be used as a starting point for your configuration.
- **Sample data** – The spreadsheet contains data that can be used as an example for business-specific data. This data can be imported and used as an example, but it must be changed for individual business practices.
- **No data** – The spreadsheet doesn't contain any data. Several areas of the product are unique to each business and its business practices. These areas must be configured specifically for the organization. These spreadsheets should be reviewed and updated for the organization as appropriate.

For more information about the type of data that is included in each entity spreadsheet in the data packages see the Data packages section of this topic. You can modify individual spreadsheets before you import the data packages, or you can import the data packages as they have been supplied and then update your data in the system.

**Using configuration data packages**

You can access configuration data packages from LCS. You can either apply them to an LCS environment, or download them so that you can manually import them.

1. Open your LCS project, and open the Asset library.
2. In the list of asset types, select Process data package.
3. Click Import.
4. Select the configuration data package.
5. Click Pick.

At this point, you can use the Consume function to apply the process data package to an LCS environment.

You can also download the individual data package files from the Data package area. Use the Data management workspace to import the data packages from LCS. For more information about how to import and export configurations, see Copy configuration data between companies or legal entities overview.

**Special considerations**
System setup
The System data package must be imported before any other data package. By default, the System data package creates a new legal entity that is named **ST01**. The data packages for the module areas depend on this legal entity.

General ledger
A generic chart of accounts is included in the configuration data packages. When this data is used as it’s defined in the Main account entity spreadsheet, posting profiles across the system are filled with default posting data. If you change the main accounts that are used for the chart of accounts, you must also update the individual posting profiles and posting accounts for each area.

Data packages: System

**010 – System Setup**

<table>
<thead>
<tr>
<th>Entity spreadsheet</th>
<th>SPREADSHEET CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business data</td>
<td>Sample data</td>
</tr>
<tr>
<td>Address and contact information purpose</td>
<td>X</td>
</tr>
<tr>
<td>Address books</td>
<td>X</td>
</tr>
<tr>
<td>Address format lines</td>
<td>X</td>
</tr>
<tr>
<td>Address format</td>
<td>X</td>
</tr>
<tr>
<td>Address parameters</td>
<td>X</td>
</tr>
<tr>
<td>Calendar</td>
<td>X</td>
</tr>
<tr>
<td>Cities</td>
<td>X</td>
</tr>
<tr>
<td>Counties</td>
<td>X</td>
</tr>
<tr>
<td>CountryRegions</td>
<td>X</td>
</tr>
<tr>
<td>Currencies</td>
<td>X</td>
</tr>
<tr>
<td>Districts</td>
<td>X</td>
</tr>
<tr>
<td>Exchange rates</td>
<td>X</td>
</tr>
<tr>
<td>Global address book parameters</td>
<td>X</td>
</tr>
<tr>
<td>Global address book</td>
<td>X</td>
</tr>
<tr>
<td>Legal entities</td>
<td>X</td>
</tr>
<tr>
<td>Name affixes</td>
<td>X</td>
</tr>
</tbody>
</table>
### Data packages: Financials

#### 020 – GL Shared

<table>
<thead>
<tr>
<th>SPREADSHEET CONTENT</th>
<th>Business data</th>
<th>Sample data</th>
<th>No data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity spreadsheet</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Account structure activation</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>SPREADSHEET CONTENT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Account structure allowed values</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Account structures</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced rule criteria</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced rule structure activation</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced rule structure allowed values</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced rule structures</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced rules</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chart of accounts</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consolidation groups and accounts</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dimension attribute activation</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Financial dimension format</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Financial dimension sets</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Financial dimension translations</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Financial dimension value translations</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Financial dimension values</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Financial dimensions</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fiscal calendar period</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fiscal calendar</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Main account categories</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Main account</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number sequence code</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number sequence group</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Entity spreadsheet</strong></td>
<td><strong>Business data</strong></td>
<td><strong>Sample data</strong></td>
<td><strong>No data</strong></td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------</td>
<td>----------------</td>
<td>------------</td>
</tr>
<tr>
<td>Accounts for automatic transactions</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Balance control accounts</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Calendar</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Date intervals</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default descriptions</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Dimension parameters</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document types</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Financial dimension value legal entity overrides</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Financial reasons</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Journal descriptions</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Journal names</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ledger allocation basis source</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Ledger allocation basis</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Ledger allocation rule destination</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Ledger allocation rule source</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Ledger allocation rule</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Ledger fiscal calendar period</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Ledger fiscal calendar year</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Ledger parameters</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPREADSHEET CONTENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ledger</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Main account legal entity overrides</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organization email template message</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organization email template</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subledger journal transfer rule</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Workflow organization parameters</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Working times</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>100 – Bank</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entity spreadsheet</td>
<td>SPREADSHEET CONTENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank accounts</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank facility groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank facility types</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank parameters</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank posting profiles</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank statement import methods for general journal</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank transaction groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank transaction type</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Check layout</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer charge groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payment purpose codes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reconciliation matching rule sets</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPREADSHEET CONTENT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reconciliation matching rules</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transaction code mapping</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 120 – AP

<table>
<thead>
<tr>
<th>SPREADSHEET CONTENT</th>
<th>Business data</th>
<th>Sample data</th>
<th>No data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity spreadsheet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sample data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aging period definitions</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Business segments</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Business subsegments</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Buyer groups</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Cash discount</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Delivery charges groups</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Destination code</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Expedite codes</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Item charge groups</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Line discount vendor groups</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Line of business</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modes of delivery</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiline discount vendor groups</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Payment calendar rule</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payment calendar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payment day lines</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payment schedule lines</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payment schedule</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Price vendor groups</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>SPREADSHEET CONTENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>----------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Procurement charge codes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product categories</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product category hierarchies</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product category hierarchy roles</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product category hierarchy translations</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Purchase pools</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales tax groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terms of delivery</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terms of payment</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total discount vendor groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor bank accounts</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor charges group</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor exception groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor invoice form printing configurations</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor invoice policy rule types</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor ledger accounts</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor parameters</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor payment fee</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor payment format</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor payment method specification</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor payment method</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPREADSHEET CONTENT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor posting profile</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor price tolerance groups</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Vendor reasons</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor, form parameters</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Vendors</td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

### 130 – Tax

<table>
<thead>
<tr>
<th>SPREADSHEET CONTENT</th>
<th>Business data</th>
<th>Sample data</th>
<th>No data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity spreadsheet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales tax authorities</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales tax code limits</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Sales tax code values</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Sales tax codes</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Sales tax exempt code</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Sales tax exempt numbers</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Sales tax group details</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales tax groups</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Sales tax item groups</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Sales tax ledger posting groups</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Sales tax parameters preset entity</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Sales tax parameters</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Sales tax period setup</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales tax reporting codes</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Transaction codes</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Withholding tax code values</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Entity spreadsheet</td>
<td>Business data</td>
<td>Sample data</td>
<td>No data</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>Aging period definitions</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business segments</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business subsegments</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cash discount</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collection letter form printing configurations</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Collection letter setup</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commission calculation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commission customer group</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Commission item group</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commission sales group</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Customer account statement form printing configurations</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Customer bank accounts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer charge groups</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Customer definitions</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Customer details</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Customer facing form printing configurations</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td><strong>SPREADSHEET CONTENT</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer ledger accounts</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer parameters</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer payment fee</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer payment method</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>specification</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer payment method</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer posting profiles</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer reasons</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer write-off reason</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>codes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customers</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delivery charges groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expedite codes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Free text invoice form</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>printing configurations</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interest codes with fees</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interest codes with ranges</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interest codes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interest note form printing</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>configurations</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item charge groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line discount customer groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line of business</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modes of delivery</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payment calendar</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payment day lines</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPREADSHEET CONTENT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>----</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payment schedule lines</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payment schedule</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Price customer groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Printed form notes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales charge codes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales districts</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales order pools</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Statistics group</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terms of delivery</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terms of payment</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total discount customer groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Configuration data templates are predefined lists of entities for each module area that can be used in a data project. You can create, view, and modify these templates by using the Template page in the Data management workspace.

**IMPORTANT**
Default configuration templates available out-of-the-box will always have the latest version of an entity. Templates can be created from an existing data project as needed.

Create a new configuration data template

The Template page in the Data management workspace provides tools that let you create a template of entities. This page resembles the configurations page, and the two features work in a similar manner. You must use Enhanced view to take advantage of the new features.

To create a template, follow these steps.

1. Select New to create a template. Enter an ID and name for the template. Notice that the status is set to Draft.
2. Add or remove entities as you require.
3. Organize the list by using the Sort by button to reorder your entities by entity group, or by unit, level, and sequence.
4. To change the sequence of any of the entities, manually edit the unit, level, or sequence. Alternatively, use the Resequence button to update any entities that you’ve selected. The Resequence button appears only if you select more than one entity.
5. To add filters to an entity, use the Filter button. Then review the results of the filters by using the Preview button. If you add a filter, the Filter button is changed to an Edit button.
6. If you don’t want all fields to be mapped, you can use the View map button to exclude fields from the mapping.
7. Select Validate template to change the status to Validated.

Your template can now be used in a project. However, you might want to use some additional features to control the export process.

- Use the Open in Excel button to open the contents of the grid in a Microsoft Excel workbook. Modify the entities as you require, and then select Publish to upload the changes back.
- If you exported a template and want to bring that template back, select Import template, browse to the template file, and then select Create template to load it.
- To replace the contents of an open template, select Replace from template, browse to the template file that has the entities to import, and then select Create template to load the template file. The values in the open template will be overwritten.
- To create a template from a project, select New to create a template. Enter an ID and name for the template, and then select Replace template from project. In the list of projects that appears, select a project, and then select Create template to bring the project entities from that project into the open template. The values in the open template will be overwritten.
Default data templates

In July 2017 update, we released predefined templates to help you create configuration data projects. The templates are sequenced, so that the data that the entities generate will be processed in the correct order. Our predefined templates are also designed to maintain the correct sequence when more than one template is added to the same data project. For more information, see the "Sequencing in the default templates" section.

Default templates are delivered together with each new release. Our long-term goal is to provide the templates in Microsoft Dynamics Lifecycle Services (LCS), so that you can push them to an instance. However, for the current releases, select the Templates tile in the Data management workspace, and then select Load default templates to load the templates. To see the Load default templates menu, you must use Enhanced view.

After the templates are loaded, you can change them to suit your business requirements. If you ever want to retrieve the original default templates, you can use the Load default templates button to add them back to your system. The templates will then be replaced with the latest versions. If you’ve made changes to the templates, you can make a copy of the old templates by exporting them.

Note that system administrator access is required in order to load default templates and import templates. This requirement helps guarantee that all entities are correctly loaded into the template.

How entities are sequenced for processing

Whether you’re creating your own templates or using the default templates, it’s important that you understand how templates are sequenced for processing during export and import.

Units, levels, and sequences

The unit, level, and sequence of an entity are used to control the order that the data is exported or imported in.

- Entities that have different units are processed in parallel.
- In the same unit, entities are processed in parallel if they have the same level.
- In the same level, entities are processed according to their sequence order in the level.
- After one level has been processed, the next level is processed.
- We use only unit 1 to help guarantee that all dependencies are handled in the correct order.

Entities are categorized by module name, entity category, and tag.

- The module represents the module that the entity is typically used for.
- The entity category represents the type of information in the category. For example, a category might include parameters or reference data.
- Tags provide additional details about the function of the entity in the feature area. For example, entities for the general ledger have General ledger as their module name. There are several tasks within the general ledger, and the tags represent those tasks. For example, Setup and Journals are tags that represent tasks that can be done for the general ledger.

Sequencing in the default templates

Our long-term goal for sequencing is to automatically sequence all the entities for every configuration. Until we reach that goal, for the entities in each of our default templates, we have created sequences that represent the dependency order between entities.

The following table shows how the templates were set up to handle dependencies. Note that the entities do not have to be processed in the order that we have sequenced them in. You can sequence them differently. However, you might inadvertently change the order that entities are processed in. If an entity requires data that hasn’t been imported by another entity, you might receive errors because of missing dependent data.
<table>
<thead>
<tr>
<th>MODULE</th>
<th>UNIT</th>
<th>LEVEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>System setup</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Global address book</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>General ledger shared</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>Workflow</td>
<td>1</td>
<td>22</td>
</tr>
<tr>
<td>General ledger</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>Band 1 for dependencies</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>Band 2 for dependencies</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>Band 3 for dependencies</td>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td>Band 4 for dependencies</td>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td>Band 5 for dependencies</td>
<td>1</td>
<td>70</td>
</tr>
<tr>
<td>Band 6 for dependencies</td>
<td>1</td>
<td>80</td>
</tr>
<tr>
<td>Band 7 for dependencies</td>
<td>1</td>
<td>90</td>
</tr>
<tr>
<td>Bank</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>Accounts payable</td>
<td>1</td>
<td>120</td>
</tr>
<tr>
<td>Tax</td>
<td>1</td>
<td>130</td>
</tr>
<tr>
<td>Accounts receivable</td>
<td>1</td>
<td>140</td>
</tr>
<tr>
<td>Fixed assets</td>
<td>1</td>
<td>150</td>
</tr>
<tr>
<td>Budgeting</td>
<td>1</td>
<td>160</td>
</tr>
<tr>
<td>Inventory management</td>
<td>1</td>
<td>300</td>
</tr>
<tr>
<td>Product management</td>
<td>1</td>
<td>310</td>
</tr>
<tr>
<td>Procurement</td>
<td>1</td>
<td>320</td>
</tr>
<tr>
<td>Sales and Marketing</td>
<td>1</td>
<td>330</td>
</tr>
<tr>
<td>Quality management</td>
<td>1</td>
<td>395</td>
</tr>
<tr>
<td>Warehouse management</td>
<td>1</td>
<td>400</td>
</tr>
<tr>
<td>Transportation management</td>
<td>1</td>
<td>405</td>
</tr>
<tr>
<td>MODULE</td>
<td>UNIT</td>
<td>LEVEL</td>
</tr>
<tr>
<td>----------------------</td>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>Production control</td>
<td>1</td>
<td>410</td>
</tr>
<tr>
<td>Process manufacturing</td>
<td>1</td>
<td>412</td>
</tr>
<tr>
<td>Costing</td>
<td>1</td>
<td>420</td>
</tr>
<tr>
<td>Retail (See the note.)</td>
<td>1</td>
<td>500</td>
</tr>
<tr>
<td>Expense management</td>
<td>1</td>
<td>600</td>
</tr>
<tr>
<td>Project accounting</td>
<td>1</td>
<td>650</td>
</tr>
</tbody>
</table>

**NOTE**

The Retail template is scheduled to be released in Finance and Operations, App update 3.

We reserved levels 10 through 22 for shared system entities, so that those entities are processed first. Almost all systems also use the company-specific general ledger entities. Therefore, we reserved level 25 for those entities. These levels represent the minimum basic setup that is required for most shared data in a configuration.

After the basic setup is completed, many entities can be loaded in parallel across all the modules. These entities don't have to be loaded in silos by module. Instead, we set up bands of dependencies between the data for different entities. We added the entities that have no dependencies to band 30. We then added band 40 for entities that have a dependency on the entities in band 30. We continued the process for bands 50 through 90.

After we organized the basic entities so that they can be processed in parallel, we organized the remaining entities by module, in the order that the modules should be processed in. However, many entities have many dependencies, some of which are complex. For example, the Vendor posting profiles entity might require Vendors or Items entities. Although the Vendor posting profiles entity is in the Accounts payable module, it must be processed after the Product management module. In that case, if Vendors entities are 1.130.10 and Items entities are 1.300.10, the Vendor posting profiles entity must be moved so that it's after that sequence (for example, 1.310.20).

The sequences that we have implemented are a guideline, not a requirement. There is no required relationship between a level and a module. You can rearrange the entities if the sequence doesn't work for your implementation. To add your own templates to a configuration, you can follow the preceding guidelines to help guarantee that your template is correctly merged into a project that uses default templates.

**Templates that have the same entity**

Some entities are required in more than one template. For example, you must have payment terms in both the Accounts Payable and Accounts Receivable templates. However, you might require only the Accounts Receivable template. We added the entity to both templates for situations where you require only one of them.

A data project can include only one instance of an entity. If you add a template, and the template contains an entity that already exists in a data project, the entity in that template replaces the entity that is currently in the project.

You can use this capability to override the default templates without changing them. For example, the worker field hasn't been mapped in your data project, but you have your own template that adds workers. In this case, you can build a template that includes the entities that have the worker field. In that template, you can map the worker field. Any entities in the data project that don't have the field mapped will then be replaced.

**Merged templates**
We have created larger templates that cover multiple module areas. You can use the larger templates or any combination of smaller templates to build a data project. The following combined templates are available:

- System and Shared, which include system setup, global address book, shared general ledger, and workflow
- Financials, which includes general ledger, bank, accounts payable, tax, accounts receivable, fixed assets, and budgeting
- Supply Chain Management, which includes inventory management, product management, procurement, sales and marketing, limited warehouse management, production control, and costing

The Expense and Project Management templates aren't included in a larger template. However, they are designed so that they can easily be merged into a project that uses other templates.

The Workers template includes the entities needed to add workers and re-map entities where the worker mapping was removed.

**Master data**

Many default templates include entities for master data such as customers, vendors, and released products. These entities are included to indicate the correct sequence of entities that you will require after you've loaded parameters and reference data. Master entities are most often sequenced in the module bands that are numbered 100 and above. In the grid, the entity category for these entities will be **Master**.

If you don't want to include master data in your configuration, remove those entities from your project.
Enable change tracking for entities

Change tracking enables incremental export of data from Finance and Operations apps by using Data management. In an incremental export, only records that have changed are exported. To enable incremental export, you must enable change tracking on entities. If you don't enable change tracking on an entity, you can only enable a full export each time.

Change tracking can be enabled for both bring your own database (BYOD) and non-BYOD scenarios. This includes retrieving record changes through Dataverse virtual entities.

**NOTE**

Change tracking will track record deletion only for bring your own database (BYOD) and Dataverse virtual entity use cases, if the entity supports it. Other non-BYOD scenarios will not include tracking record deletion. Deletion is tracked only for the root data source in the entity.

Enable change tracking for BYOD

You can enable change tracking when you publish one or more entities to a data store (BYOD).

1. In the Data management workspace, select **Configure entity export to database**.

2. Select the database to export data to, and then select **Publish**.

   You can publish one or more entities to your database. Select **Show published only** to see a list of entities that have previously been published.

3. Select an entity that is published, and then select **Change tracking**.

4. Select the appropriate option for change tracking for your environment.

   An entity can be modeled by using more than one table. The options let you specify the granularity at which changes can be tracked in an entity.

<table>
<thead>
<tr>
<th>OPTION</th>
<th>HOW CHANGES ARE TRACKED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable primary table</td>
<td>Changes that are made to any fields in the primary table trigger a change in entity. Changes that are made to fields in secondary tables don't trigger a change in entity.</td>
</tr>
<tr>
<td>Enable entire entity</td>
<td>Changes that are made to any fields in any table in the entity trigger a change in entity.</td>
</tr>
<tr>
<td>Enable custom query</td>
<td>Uses a custom query that identifies the tables on which changes must be tracked. The custom query is defined in the entity.</td>
</tr>
</tbody>
</table>
Enable change tracking for non-BYOD scenarios

Change tracking can be enabled for non-BYOD scenarios. This includes retrieving record changes through Dataverse virtual entities for Finance and Operations apps. When change tracking is enabled for an entity, changes can be retrieved through the entity’s OData endpoint by adding `odata.track-changes` as a preference header.

For more information on using change tracking for an entity, see Use change tracking to synchronize data with external systems.

To enable change tracking for non-BYOD scenarios:

1. From the Data management workspace, select the Data entities list page.
2. Select the entity for which you want to enable change tracking.
3. Select the Change tracking action on the action ribbon, and select the desired option for how changes should be tracked for the entity. See the table in the Enable change tracking for BYOD section above for detail on the available options.

Custom query for change tracking

The following example shows how to add a static method to an entity. You must make sure that the method returns a query, and that the root node is the same as the entity. For example, for the Customer entity, the root node is custTable, and the change tracking query for it is also custtable.

- You must enable change tracking on the tables that are part of the query.
- Create a join between the entity and the change tracking query (on the root table) to determine which records have changed in the entity.

```java
public static Query defaultCTQuery()
{
    Query q = new Query();

    QueryBuildDataSource custDs = q.addDataSource(tableNum(CustTable));
    QueryBuildDataSource partyDs = custDs.addDataSource(tableNum(DirPartyTable));
    partyDs.relations(true);
    QueryBuildDataSource locationDs = partyDs.addDataSource(tableNum(DirPartyLocation));
    locationDs.addRange(fieldNum(DirPartyLocation, IsPrimary)).value(queryValue(NoYes::Yes));
    locationDs.addLink(fieldNum(DirPartyTable, RecId), fieldNum(DirPartyLocation, Party));
    QueryBuildDataSource addressDs = locationDs.addDataSource(tableStr(LogisticsPostalAddress));
    addressDs.addLink(fieldNum(DirPartyLocation, Location), fieldNum(LogisticsPostalAddress, Location));

    return q;
}
```

NOTE

If a change is triggered, the change is tracked on the entire record and not at the field level. The entire entity record is exported to the destination. Regardless of the option that you select, the number of fields in the entity is the number that is exported to the destination.
The application ships with many default data entities. Data entities are frequently updated, so for documentation, we rely on the data entity templates to indicate which order data entities should be imported in, and on reports for a list of data entities that ship with each release.

**Configuration data packages**

Configuration data packages on Microsoft Dynamics Lifecycle Services (LCS) contain configuration entity spreadsheets. Configuration entity spreadsheets contain best practice data that you can use to create an initial golden build of an implementation. The data entities in the data packages are also sequenced appropriately using an XML file to help guarantee a successful single-click import of the data. We recommend that you download and review the configuration data packages to understand how we recommend that you order your data imports. For more information, see [Configuration data templates](#) and [Copy configuration data between companies or legal entities overview](#).

**Reports**

Microsoft provides the following reports for data entities, which can be downloaded from [Technical reference reports](#):

- Aggregate data entities: Lists the aggregate data entities, and the fields that each contains.
- Aggregate measures: Lists the aggregate measures.
- Config keys: Lists the configuration keys.
- Config key groups: Lists the configuration key groups.
- Data entities: Lists each data entity. The report indicates the data source of the entity and the fields included in the entity. The report also indicates whether the data entity is public.
- Data entities fields: Lists each field in a data entity, and the table that it originates from.
- KPIs: Lists the KPIs.
- License codes: Lists the license code associated with each configuration key.
- Menu items: Lists the menu items associated with each configuration key.
- SSRS reports: Lists each report. The report indicates the data set used for each report, as well as the filters and fields available on each report.
- Tables: Lists each table and its table group.
- Workflow type: Lists each type of workflow. The report also describes what each type of workflow is used for and indicates whether the workflows of each type are associated with a specific company in the organization or with the whole organization.

**Scripts**

You can download the scripts to run these reports from [fin-ops-doc-scripts](#).

**Additional resources**

[Data entities overview](#)
Data management in the application supports Microsoft Excel-based templates for data entities. These templates can contain one or more worksheets. Templates with multiple worksheets are often used when it is convenient to manage data in a single file and import it to multiple data entities. An example would be sites and warehouses.

Upload a file once and map it to all entities

Let’s take an example where there is one Excel file with worksheets called Sites and Warehouses. To set up the data import project, you would add the first data entity, Sites and then upload the file. You will be able to select Sites as the worksheet to be used for this entity.

If you add the second entity Warehouses without leaving the Add file form, the worksheet lookup will let you select the Warehouses worksheet without having to upload the file again. The only reason to upload a new file would be if the Warehouses data was in a different file.

Fix worksheet to entity mapping

The mapping of the worksheet to a data entity in the import job can be fixed from the grid. The Worksheet column in the grid shows the worksheets from the file that was mapped. You can choose a different worksheet from the drop-down menu. If the chosen worksheet is already mapped to an entity in the data project, the system asks you to confirm the change. We recommend that you fix all mappings in the grid.
Re-map to a new file

In cases where a new version of the same file or a completely new file must be uploaded for existing entities in a data project, you must use the **Add file** experience, and add the entities again as if they were being added for the first time. The system will confirm that you want to overwrite the existing entities in the data project before proceeding. Entities that are not added again (or overwritten) will continue to hold the previous mappings from the previous file.

Upload a file using Run project

You can upload an Excel file while using the **Run project** option to execute an import project. You must be careful to upload only files that have the same worksheets as the existing mappings on the data entities in the data project. If a worksheet is not found in the newly uploaded file, the system displays an error and will stop the import. If the mapping to the worksheet must be changed for an entity, then the mappings in the data project must be first updated from within the data project before using the file in the **Run project** experience.
Enable change tracking for entities

Change tracking enables incremental export of data from Finance and Operations apps by using Data management. In an incremental export, only records that have changed are exported. To enable incremental export, you must enable change tracking on entities. If you don’t enable change tracking on an entity, you can only enable a full export each time.

Change tracking can be enabled for both bring your own database (BYOD) and non-BYOD scenarios. This includes retrieving record changes through Dataverse virtual entities.

**NOTE**

Change tracking will track record deletion only for bring your own database (BYOD) and Dataverse virtual entity use cases, if the entity supports it. Other non-BYOD scenarios will not include tracking record deletion. Deletion is tracked only for the root data source in the entity.

Enable change tracking for BYOD

You can enable change tracking when you publish one or more entities to a data store (BYOD).

1. In the Data management workspace, select **Configure entity export to database**.
2. Select the database to export data to, and then select **Publish**.
   
   You can publish one or more entities to your database. Select **Show published only** to see a list of entities that have previously been published.
3. Select an entity that is published, and then select **Change tracking**.
4. Select the appropriate option for change tracking for your environment.

An entity can be modeled by using more than one table. The options let you specify the granularity at which changes can be tracked in an entity.

<table>
<thead>
<tr>
<th>OPTION</th>
<th>HOW CHANGES ARE TRACKED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable primary table</td>
<td>Changes that are made to any fields in the primary table trigger a change in entity. Changes that are made to fields in secondary tables don't trigger a change in entity.</td>
</tr>
<tr>
<td>Enable entire entity</td>
<td>Changes that are made to any fields in any table in the entity trigger a change in entity.</td>
</tr>
<tr>
<td>Enable custom query</td>
<td>Uses a custom query that identifies the tables on which changes must be tracked. The custom query is defined in the entity.</td>
</tr>
</tbody>
</table>
Enable change tracking for non-BYOD scenarios

Change tracking can be enabled for non-BYOD scenarios. This includes retrieving record changes through Dataverse virtual entities for Finance and Operations apps. When change tracking is enabled for an entity, changes can be retrieved through the entity’s OData endpoint by adding `odata.track-changes` as a preference header.

For more information on using change tracking for an entity, see Use change tracking to synchronize data with external systems.

To enable change tracking for non-BYOD scenarios:

1. From the Data management workspace, select the Data entities list page.
2. Select the entity for which you want to enable change tracking.
3. Select the Change tracking action on the action ribbon, and select the desired option for how changes should be tracked for the entity. See the table in the Enable change tracking for BYOD section above for detail on the available options.

Custom query for change tracking

The following example shows how to add a static method to an entity. You must make sure that the method returns a query, and that the root node is the same as the entity. For example, for the Customer entity, the root node is custTable, and the change tracking query for it is also custtable.

- You must enable change tracking on the tables that are part of the query.
- Create a join between the entity and the change tracking query (on the root table) to determine which records have changed in the entity.

```java
public static Query defaultCTQuery() {
    Query q = new Query();

    QueryBuildDataSource custDs = q.addDataSource(tableNum(CustTable));
    QueryBuildDataSource partyDs = custDs.addDataSource(tableNum(DirPartyTable));
    partyDs.relations(true);

    QueryBuildDataSource locationDs = partyDs.addDataSource(tableNum(DirPartyLocation));
    locationDs.addRange(fieldNum(DirPartyLocation, IsPrimary)).value(queryValue(NoYes::Yes));
    locationDs.addLink(fieldNum(DirPartyTable, RecId), fieldNum(DirPartyLocation, Party));

    QueryBuildDataSource addressDs = locationDs.addDataSource(tableStr(LogisticsPostalAddress));
    addressDs.addLink(fieldNum(DirPartyLocation, Location), fieldNum(LogisticsPostalAddress, Location));

    return q;
}
```
Configuration keys and data entities

Before you use data entities to import or export data, we recommended that you first determine the impact of configuration keys on the data entities that you are planning to use.

To learn more about configuration keys, see the License codes and configuration keys report.

Configuration key assignments

Configuration keys can be assigned to one or all of the following artifacts.

- Data entities
- Tables used as data sources
- Table fields
- Data entity fields

The following table summarizes how configuration key values on the different artifacts that underlie an object change the expected behavior of the object.

<table>
<thead>
<tr>
<th>CONFIGURATION KEY SETTING ON DATA ENTITY</th>
<th>CONFIGURATION KEY SETTING ON TABLE</th>
<th>CONFIGURATION KEY SETTING ON TABLE FIELD</th>
<th>CONFIGURATION KEY ON DATA ENTITY FIELD</th>
<th>EXPECTED BEHAVIOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disabled</td>
<td>Not evaluated</td>
<td>Not evaluated</td>
<td>Not evaluated</td>
<td>If the configuration key for the data entity is disabled, the data entity will not be functional. It does not matter whether the configuration keys in the underlying tables and fields are enabled or disabled.</td>
</tr>
<tr>
<td>CONFIGURATION KEY SETTING ON DATA ENTITY</td>
<td>CONFIGURATION KEY SETTING ON TABLE</td>
<td>CONFIGURATION KEY SETTING ON TABLE FIELD</td>
<td>CONFIGURATION KEY ON DATA ENTITY FIELD</td>
<td>EXPECTED BEHAVIOR</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------------------------------</td>
<td>----------------------------------------</td>
<td>--------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Enabled</td>
<td>Disabled</td>
<td>Not evaluated</td>
<td>Not evaluated</td>
<td>If the configuration key for a data entity is enabled, the data management framework checks the configuration key on each of the underlying tables. If the configuration key for a table is disabled, that table will not be available in the data entity for functional use. If a table's configuration key is disabled, the table and data entity configuration key settings are not evaluated. If the primary table in the entity has its configuration key disabled, then the system will act as though the entity's configuration key were disabled.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Enabled</td>
<td>Disabled</td>
<td>Not evaluated</td>
<td>If the configuration key for a data entity is enabled, and the underlying tables configuration keys are enabled, the data management framework will check the configuration key on of the fields in the tables. If the configuration key for a field is disabled, that field will not be available in the data entity for functional use even if the corresponding data entity field has the configuration key enabled.</td>
</tr>
</tbody>
</table>
If an entity has another entity as a data source then, the above semantics are applied in a recursive manner.

**Entity list refresh**

When the entity list is refreshed, the data management framework builds the configuration key metadata for runtime use. This metadata is built using the logic described above. We strongly recommend that you wait for the entity list refresh to complete before using jobs and entities in the data management framework. If you don’t wait, the configuration key metadata may not be up to date and could result in unexpected outcomes. When the entity list is being refreshed, the following message is shown in the entity list page.

**Data entity list page**

The data entity list page in the Data management workspace shows the configuration key settings for the entities. Start from this page to understand the impact from configuration keys on the data entity.

This information is shown using the metadata that is built during entity refresh. The configuration key column shows the name of the configuration key that is associated with the data entity. If this column is blank it means that there is no configuration key associated with the data entity. The configuration key status column shows the state of the configuration key. If it has a checkmark, it means the key is enabled. If it is blank, it means either the key is disabled or there is no key associated.
Target fields

The next step is to drill into the data entity to view the impact of configuration keys on tables and fields. The target fields form for a data entity shows configuration key and the key status information for the related tables and fields in the data entity. If the data entity itself has its configuration key disabled, a warning message is shown informing that the tables and fields in the target fields form for this entity will not be available at all regardless of their configuration key status.

Child entities

Certain entities have other entities as data sources, or are composite data entities: configuration key information for these entities is shown in the Child entities form. Use this form in the similar way to the entities list page described above. The target fields form for the child entity also behaves like what is described above.

Using data entities

After understanding the full impact, if any, of configuration keys on the data entities that you would like to use, you can now proceed to using the data entities by adding them to data projects.

Run time validations for configuration keys

Using the configuration key metadata built during entity refresh list, run time validations are performed in the following use cases:

- When a data entity is added to a job
When user clicks ‘validate’ on the entity list
- When the user loads a data package into a data project
- When the user loads a template into a data project
- When an existing data project is loaded
- When a template is loaded into a data project
- Before the export/import job is executed (batch, non-batch, recurring, OData)
- When the user generates mapping
- When the user maps fields in the mapping UI
- When the user adds only ‘importable fields’

Anytime that you update configuration keys at the entity, table or field level, the entity list in the data management framework must be refreshed. This process ensures that the framework picks up the latest configuration key settings. Until the entity list is refreshed, the following warning will be shown in the entity list page. The updated configuration key changes will take effect immediately after the entity list is refreshed. We recommend that you validate existing data projects and jobs to make sure that they function as expected after the configuration keys changes are put in effect.
Configuration data packages are available as process data packages from Microsoft Dynamics Lifecycle Services (LCS). These data packages can help improve the repeatability of implementations and accelerate the configuration.

Data packages contain configuration entity spreadsheets. These entity spreadsheets contain best practice data that you can use to create an initial golden build. The data entities in the data packages are also sequenced appropriately to help guarantee a successful single-click import of the data.

The entity spreadsheets include three types of data:

- **Business data** – The spreadsheet contains standard business data for a mid-sized trade or retail company. This data combines best practices and business standards that can be used as a starting point for your configuration.
- **Sample data** – The spreadsheet contains data that can be used as an example for business-specific data. This data can be imported and used as an example, but it must be changed for individual business practices.
- **No data** – The spreadsheet doesn't contain any data. Several areas of the product are unique to each business and its business practices. These areas must be configured specifically for the organization. These spreadsheets should be reviewed and updated for the organization as appropriate.

For more information about the type of data that is included in each entity spreadsheet in the data packages see the Data packages section of this topic. You can modify individual spreadsheets before you import the data packages, or you can import the data packages as they have been supplied and then update your data in the system.

### Using configuration data packages

You can access configuration data packages from LCS. You can either apply them to an LCS environment, or download them so that you can manually import them.

1. Open your LCS project, and open the Asset library.
2. In the list of asset types, select **Process data package**.
3. Click **Import**.
4. Select the configuration data package.
5. Click **Pick**.

At this point, you can use the **Consume** function to apply the process data package to an LCS environment.

You can also download the individual data package files from the **Data package** area. Use the **Data management** workspace to import the data packages from LCS. For more information about how to import and export configurations, see Copy configuration data between companies or legal entities overview.

### Special considerations
System setup

The System data package must be imported before any other data package. By default, the System data package creates a new legal entity that is named ST01. The data packages for the module areas depend on this legal entity.

General ledger

A generic chart of accounts is included in the configuration data packages. When this data is used as it’s defined in the Main account entity spreadsheet, posting profiles across the system are filled with default posting data. If you change the main accounts that are used for the chart of accounts, you must also update the individual posting profiles and posting accounts for each area.

Data packages: System

**010 – System Setup**

<table>
<thead>
<tr>
<th>Entity spreadsheet</th>
<th>SPREADSHEET CONTENT</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity spreadsheet</td>
<td>Business data</td>
<td>Sample data</td>
<td>No data</td>
<td></td>
</tr>
<tr>
<td>Address and contact information purpose</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address books</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address format lines</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address format</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address parameters</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Calendar</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cities</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Counties</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CountryRegions</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Currencies</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Districts</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exchange rates</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Global address book parameters</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Global address book</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Legal entities</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name affixes</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SPREADSHEET CONTENT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name sequences</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating unit</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organization hierarchy purposes</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organization hierarchy type</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organization hierarchy</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Party contacts</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Party postal addresses</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Party relationships</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Postal codes</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relationship types</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>States</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System parameters</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team types</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teams</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unit conversions</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unit translations</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Units</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User groups</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User information</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Data packages: Financials**

**020 – GL Shared**

<table>
<thead>
<tr>
<th><strong>SPREADSHEET CONTENT</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity spreadsheet</td>
<td>Business data</td>
</tr>
<tr>
<td>Account structure activation</td>
<td>X</td>
</tr>
<tr>
<td>SPREADSHEET CONTENT</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>---</td>
</tr>
<tr>
<td>Account structure allowed values</td>
<td>X</td>
</tr>
<tr>
<td>Account structures</td>
<td>X</td>
</tr>
<tr>
<td>Advanced rule criteria</td>
<td>X</td>
</tr>
<tr>
<td>Advanced rule structure activation</td>
<td>X</td>
</tr>
<tr>
<td>Advanced rule structure allowed values</td>
<td>X</td>
</tr>
<tr>
<td>Advanced rule structures</td>
<td>X</td>
</tr>
<tr>
<td>Advanced rules</td>
<td>X</td>
</tr>
<tr>
<td>Chart of accounts</td>
<td>X</td>
</tr>
<tr>
<td>Consolidation groups and accounts</td>
<td>X</td>
</tr>
<tr>
<td>Dimension attribute activation</td>
<td>X</td>
</tr>
<tr>
<td>Financial dimension format</td>
<td>X</td>
</tr>
<tr>
<td>Financial dimension sets</td>
<td>X</td>
</tr>
<tr>
<td>Financial dimension translations</td>
<td>X</td>
</tr>
<tr>
<td>Financial dimension value translations</td>
<td>X</td>
</tr>
<tr>
<td>Financial dimension values</td>
<td>X</td>
</tr>
<tr>
<td>Financial dimensions</td>
<td>X</td>
</tr>
<tr>
<td>Fiscal calendar period</td>
<td>X</td>
</tr>
<tr>
<td>Fiscal calendar</td>
<td>X</td>
</tr>
<tr>
<td>Main account categories</td>
<td>X</td>
</tr>
<tr>
<td>Main account</td>
<td>X</td>
</tr>
<tr>
<td>Number sequence code</td>
<td>X</td>
</tr>
<tr>
<td>Number sequence group</td>
<td>X</td>
</tr>
<tr>
<td>Entity spreadsheet</td>
<td>Business data</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Accounts for automatic transactions</td>
<td>X</td>
</tr>
<tr>
<td>Balance control accounts</td>
<td></td>
</tr>
<tr>
<td>Calendar</td>
<td></td>
</tr>
<tr>
<td>Date intervals</td>
<td></td>
</tr>
<tr>
<td>Default descriptions</td>
<td></td>
</tr>
<tr>
<td>Dimension parameters</td>
<td>X</td>
</tr>
<tr>
<td>Document types</td>
<td>X</td>
</tr>
<tr>
<td>Financial dimension value legal entity overrides</td>
<td></td>
</tr>
<tr>
<td>Financial reasons</td>
<td>X</td>
</tr>
<tr>
<td>Journal descriptions</td>
<td>X</td>
</tr>
<tr>
<td>Journal names</td>
<td>X</td>
</tr>
<tr>
<td>Ledger allocation basis source</td>
<td>X</td>
</tr>
<tr>
<td>Ledger allocation basis</td>
<td></td>
</tr>
<tr>
<td>Ledger allocation rule destination</td>
<td></td>
</tr>
<tr>
<td>Ledger allocation rule source</td>
<td></td>
</tr>
<tr>
<td>Ledger allocation rule</td>
<td></td>
</tr>
<tr>
<td>Ledger fiscal calendar period</td>
<td></td>
</tr>
<tr>
<td>Ledger fiscal calendar year</td>
<td></td>
</tr>
<tr>
<td>Ledger parameters</td>
<td>X</td>
</tr>
<tr>
<td>Ledger</td>
<td>X</td>
</tr>
<tr>
<td>--------</td>
<td>---</td>
</tr>
<tr>
<td>Main account legal entity overrides</td>
<td>X</td>
</tr>
<tr>
<td>Organization email template message</td>
<td>X</td>
</tr>
<tr>
<td>Organization email template</td>
<td>X</td>
</tr>
<tr>
<td>Subledger journal transfer rule</td>
<td>X</td>
</tr>
<tr>
<td>Workflow organization parameters</td>
<td>X</td>
</tr>
<tr>
<td>Working times</td>
<td>X</td>
</tr>
</tbody>
</table>

### 100 – Bank

<table>
<thead>
<tr>
<th>SPREADSHEET CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entity spreadsheet</strong></td>
</tr>
<tr>
<td>Bank accounts</td>
</tr>
<tr>
<td>Bank facility groups</td>
</tr>
<tr>
<td>Bank facility types</td>
</tr>
<tr>
<td>Bank groups</td>
</tr>
<tr>
<td>Bank parameters</td>
</tr>
<tr>
<td>Bank posting profiles</td>
</tr>
<tr>
<td>Bank statement import methods for general journal</td>
</tr>
<tr>
<td>Bank transaction groups</td>
</tr>
<tr>
<td>Bank transaction type</td>
</tr>
<tr>
<td>Check layout</td>
</tr>
<tr>
<td>Customer charge groups</td>
</tr>
<tr>
<td>Payment purpose codes</td>
</tr>
<tr>
<td>Reconciliation matching rule sets</td>
</tr>
<tr>
<td>Reconciliation matching rules</td>
</tr>
<tr>
<td>Transaction code mapping</td>
</tr>
</tbody>
</table>

**120 – AP**

<table>
<thead>
<tr>
<th>Entity spreadsheet</th>
<th>Business data</th>
<th>Sample data</th>
<th>No data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aging period definitions</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Business segments</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Business subsegments</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Buyer groups</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Cash discount</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Delivery charges groups</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Destination code</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Expedite codes</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Item charge groups</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Line discount vendor groups</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Line of business</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Modes of delivery</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Multiline discount vendor groups</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Payment calendar rule</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Payment calendar</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Payment day lines</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Payment schedule lines</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Payment schedule</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Price vendor groups</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>SPREADSHEET CONTENT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Procurement charge codes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product categories</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product category hierarchies</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product category hierarchy roles</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product category hierarchy translations</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Purchase pools</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales tax groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terms of delivery</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terms of payment</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total discount vendor groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor bank accounts</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor charges group</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor exception groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor exception groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor invoice form printing configurations</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor invoice policy rule types</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor ledger accounts</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor parameters</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor payment fee</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor payment format</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor payment method specification</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor payment method</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Vendor posting profile

<table>
<thead>
<tr>
<th>SPREADSHEET CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor posting profile</td>
</tr>
</tbody>
</table>

### Vendor price tolerance groups

<table>
<thead>
<tr>
<th>SPREADSHEET CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor price tolerance groups</td>
</tr>
</tbody>
</table>

### Vendor reasons

<table>
<thead>
<tr>
<th>SPREADSHEET CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor reasons</td>
</tr>
</tbody>
</table>

### Vendor, form parameters

<table>
<thead>
<tr>
<th>SPREADSHEET CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor, form parameters</td>
</tr>
</tbody>
</table>

### Vendors

<table>
<thead>
<tr>
<th>SPREADSHEET CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendors</td>
</tr>
</tbody>
</table>

### 130 – Tax

<table>
<thead>
<tr>
<th>SPREADSHEET CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity spreadsheet</td>
</tr>
<tr>
<td>Sales tax authorities</td>
</tr>
<tr>
<td>Sales tax code limits</td>
</tr>
<tr>
<td>Sales tax code values</td>
</tr>
<tr>
<td>Sales tax codes</td>
</tr>
<tr>
<td>Sales tax exempt code</td>
</tr>
<tr>
<td>Sales tax exempt numbers</td>
</tr>
<tr>
<td>Sales tax group details</td>
</tr>
<tr>
<td>Sales tax groups</td>
</tr>
<tr>
<td>Sales tax item groups</td>
</tr>
<tr>
<td>Sales tax ledger posting groups</td>
</tr>
<tr>
<td>Sales tax parameters preset entity</td>
</tr>
<tr>
<td>Sales tax parameters</td>
</tr>
<tr>
<td>Sales tax period setup</td>
</tr>
<tr>
<td>Sales tax reporting codes</td>
</tr>
<tr>
<td>Transaction codes</td>
</tr>
<tr>
<td>Withholding tax code values</td>
</tr>
</tbody>
</table>
### Withholding tax codes

<table>
<thead>
<tr>
<th>SPREADSHEET CONTENT</th>
<th>X</th>
</tr>
</thead>
</table>

### Withholding tax group details

<table>
<thead>
<tr>
<th>SPREADSHEET CONTENT</th>
<th>X</th>
</tr>
</thead>
</table>

### Withholding tax groups

<table>
<thead>
<tr>
<th>SPREADSHEET CONTENT</th>
<th>X</th>
</tr>
</thead>
</table>

### Withholding tax limits

<table>
<thead>
<tr>
<th>SPREADSHEET CONTENT</th>
<th>X</th>
</tr>
</thead>
</table>

### 140 – AR

<table>
<thead>
<tr>
<th>Entity spreadsheet</th>
<th>Business data</th>
<th>Sample data</th>
<th>No data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aging period definitions</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business segments</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Business subsegments</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Cash discount</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Collection letter form printing configurations</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Collection letter setup</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Commission calculation</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Commission customer group</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Commission item group</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Commission sales group</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Customer account statement form printing configurations</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Customer bank accounts</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Customer charge groups</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Customer definitions</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Customer details</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Customer facing form printing configurations</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>SPREADSHEET CONTENT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer ledger accounts</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer parameters</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer payment fee</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer payment method specification</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer payment method</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer posting profiles</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer reasons</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer write-off reason codes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customers</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delivery charges groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expedite codes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Free text invoice form printing configurations</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interest codes with fees</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interest codes with ranges</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interest codes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interest note form printing configurations</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item charge groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line discount customer groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line of business</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modes of delivery</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payment calendar</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payment day lines</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SPREADSHEET CONTENT</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-----</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payment schedule lines</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payment schedule</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Price customer groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Printed form notes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales charge codes</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales districts</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales order pools</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Statistics group</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terms of delivery</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terms of payment</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total discount customer groups</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
To create and manage data import and export jobs, you use the Data management workspace. By default, the data import and export process creates a staging table for each entity in the target database. Staging tables let you verify, clean up, or convert data before you move it.

**NOTE**
This topic assumes that you are familiar with data entities.

### Data import/export process

Here are the steps to import or export data.

1. Create an import or export job where you complete the following tasks:
   - Define the project category.
   - Identify the entities to import or export.
   - Set the data format for the job.
   - Sequence the entities, so that they are processed in logical groups and in an order that makes sense.
   - Determine whether to use staging tables.
2. Validate that the source data and target data are mapped correctly.
3. Verify the security for your import or export job.
4. Run the import or export job.
5. Validate that the job ran as expected by reviewing the job history.
6. Clean up the staging tables.

The remaining sections of this topic provide more details about each step of the process.

**NOTE**
In order to refresh the Data import/export form to see the latest progress, use the form refresh icon. Browser level refresh is not recommended because it will interrupt any import/export jobs that are not run in batch.

### Create an import or export job

A data import or export job can be run one time or many times.

**Define the project category**

We recommend that you take the time to select an appropriate project category for your import or export job. Project categories can help you manage related jobs.

**Identify the entities to import or export**

You can add specific entities to an import or export job or select a template to apply. Templates fill a job with a list of entities. The **Apply template** option is available after you give the job a name and save the job.
Set the data format for the job

When you select an entity, you must select the format of the data that will be exported or imported. You define formats by using the Data sources setup tile. A source data format is a combination of Type, File format, Row delimiter and Column delimiter. There are also other attributes, but these are the key ones to understand. The following table lists the valid combinations.

<table>
<thead>
<tr>
<th>FILE FORMAT</th>
<th>ROW/COLUMN DELIMITER</th>
<th>XML STYLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Excel</td>
<td>Excel</td>
<td>-NA-</td>
</tr>
<tr>
<td>XML</td>
<td>-NA-</td>
<td>XML-Element XML-Attribute</td>
</tr>
<tr>
<td>Delimited, fixed width</td>
<td>Comma, semicolon, tab, vertical bar, colon</td>
<td>-NA-</td>
</tr>
</tbody>
</table>

**NOTE**

It is important to select the correct value for Row delimiter, Column delimiter, and Text qualifier, if the File format option is set to Delimited. Make sure that your data doesn't contain the character used as delimiter or qualifier, as this may result in errors during import and export.

Sequence the entities

Entities can be sequenced in a data template, or in import and export jobs. When you run a job that contains more than one data entity, you must make sure that the data entities are correctly sequenced. You sequence entities primarily so that you can address any functional dependencies among entities. If entities don’t have any functional dependencies, they can be scheduled for parallel import or export.

Execution units, levels, and sequences

The execution unit, level in the execution unit, and sequence of an entity help control the order that the data is exported or imported in.

- Entities in different execution units are processed in parallel.
- In each execution unit, entities are processed in parallel if they have the same level.
- In each level, entities are processed according to their sequence number in that level.
- After one level has been processed, the next level is processed.

Resequencing

You might want to resequence your entities in the following situations:

- If only one data job is used for all your changes, you can use resequencing options to optimize the execution time for the full job. In these cases, you can use the execution unit to represent the module, the level to represent the feature area in the module, and the sequence to represent the entity. By using this approach, you can work across modules in parallel, but you can still work in sequence in a module. To help guarantee that parallel operations succeed, you must consider all dependencies.
- If multiple data jobs are used (for example, one job for each module), you can use sequencing to affect the level and sequence of entities for optimal execution.
- If there are no dependencies at all, you can sequence entities at different execution units for maximum optimization.

The Resequencing menu is available when multiple entities are selected. You can resequence based on execution unit, level, or sequence options. You can set an increment to resequence the entities that have been selected. The unit, level, and/or sequence number that is selected for each entity is updated by the specified increment.
Sorting
Use can use the Sort by option to view the entity list in sequential order.

Truncating
For import projects, you can choose to truncate records in the entities prior to import. This is useful if your records must be imported into a clean set of tables. This setting is off by default.

Validate that the source data and target data are mapped correctly
Mapping is a function that applies to both import and export jobs.

- In the context of an import job, mapping describes which columns in the source file become the columns in the staging table. Therefore, the system can determine which column data in the source file must be copied into which column of the staging table.
- In the context of an export job, mapping describes which columns of the staging table (that is, the source) become the columns in the target file.

If the column names in the staging table and the file match, the system automatically establishes the mapping, based on the names. However, if the names differ, columns aren’t mapped automatically. In these cases, you must complete the mapping by selecting the View map option on the entity in the data job.

There are two mapping views: Mapping visualization, which is the default view, and Mapping details. A red asterisk (*) identifies any required fields in the entity. These fields must be mapped before you can work with the entity. You can unmap other fields as you require when you work with the entity. To unmap a field, select the field in either the Entity column or the Source column, and then select Delete selection. Select Save to save your changes, and then close the page to return to the project. You can use the same process to edit the field mapping from source to staging after you import.

You can generate a mapping on the page by selecting Generate source mapping. A generated mapping behaves like an automatic mapping. Therefore, you must manually map any unmapped fields.

Verify the security for your import or export job
Access to the **Data management** workspace can be restricted, so that non-administrator users can access only specific data jobs. Access to a data job implies full access to the execution history of that job and access to the staging tables. Therefore, you must make sure that appropriate access controls are in place when you create a data job.

**Secure a job by roles and users**

Use the **Applicable roles** menu to restrict the job to one or more security roles. Only users in those roles will have access to the job.

You can also restrict a job to specific users. When you secure a job by users instead of roles, there is more control if multiple users are assigned to a role.

**Secure a job by legal entity**

Data jobs are global in nature. Therefore, if a data job was created and used in a legal entity, the job will be visible in other legal entities in the system. This default behavior might be preferred in some application scenarios. For example, an organization that imports invoices by using data entities might provide a centralized invoice processing team that is responsible for managing invoice errors for all divisions in the organization. In this scenario, it’s useful for the centralized invoice processing team to have access to invoice import jobs from all legal entities. Therefore, the default behavior meets the requirement from a legal entity perspective.

However, an organization might want to have invoice processing teams per legal entity. In this case, a team in a legal entity should have access only to the invoice import job in its own legal entity. To meet this requirement, you can configure legal entity–based access control on the data jobs by using the **Applicable legal entities** menu inside the data job. After the configuration is done, users can see only jobs that are available in the legal entity that they are currently signed in to. To see jobs from another legal entity, users must switch to that legal entity.

A job can be secured by roles, users, and legal entity at the same time.

**Run the import or export job**

You can run a job one time by selecting the **Import** or **Export** button after you define the job. To set up a recurring job, select **Create recurring data job**.

**NOTE**

An import or an export job can be run by selecting the **Import** or **Export** button. This will schedule a batch job to run only once. The job may not execute immediately if batch service is throttling due to the load on the batch service. The jobs can also be run synchronously by selecting **Import now** or **Export now**. This starts the job immediately and is useful if the batch does not start due to throttling. The jobs can also be scheduled to execute at a later time. This can be done by choosing the **Run in batch** option. Batch resources are subject to throttling, so the batch job might not start immediately. Using a batch is the recommended option because it will also help with large volumes of data that need to be imported or exported. Batch jobs can be scheduled to run on a specific batch group, which allows more control from a load balancing perspective.

**Validate that the job ran as expected**

The job history is available for troubleshooting and investigation on both import and export jobs. Historical job runs are organized by time ranges.
Each job run provides the following details:

- Execution details
- Execution log

Execution details show the state of each data entity that the job processed. Therefore, you can quickly find the following information:

- Which entities were processed.
- For an entity, how many records were successfully processed, and how many failed.
- The staging records for each entity.

You can download the staging data in a file for export jobs, or you can download it as a package for import and export jobs.

From the execution details, you can also open the execution log.

**Parallel imports**

To speed up the import of data, parallel processing of importing a file can be enabled if the entity supports parallel imports. To configure the parallel import for an entity, the following steps must be followed.

1. Go to System administration > Workspaces > Data management.
2. In the Import / Export section, select the Framework parameters tile to open the Data import/export framework parameters page.
3. On the Entity settings tab, select Configure entity execution parameters to open the Entity import execution parameters page.
4. Set the following fields to configure parallel import for an entity:
In the **Entity** field, select the entity.

In the **Import threshold record count** field, enter the threshold record count for import. This determines the record count to be processed by a thread. If a file has 10K records, a record count of 2500 with a task count of 4 will mean, each thread will process 2500 records.

In the **Import task count** field, enter the count of import tasks. This must not exceed the max batch threads allocated for batch processing in **System administration > Server configuration**.

### Job history clean up

The job history clean-up functionality in data management must be used to schedule a periodic cleanup of the execution history. This functionality replaces the previous staging table clean-up functionality, which is now deprecated. The following tables will be cleaned up by the clean-up process.

- All staging tables
- DMFSTAGINGVALIDATIONLOG
- DMFSTAGINGEXECUTIONERRORS
- DMFSTAGINGLOGDETAIL
- DMFSTAGINGLOG
- DMFDEFINITIONGROUPEXECUTIONHISTORY
- DMFEXECUTION
- DMFDEFINITIONGROUPEXECUTION

The **Execution history cleanup** feature must be enabled in feature management and then can be accessed from **Data management > Job history cleanup**.

### Scheduling parameters

When scheduling the clean-up process, the following parameters must be specified to define the clean-up criteria.

- **Number of days to retain history** – This setting is used to control the amount of execution history to be preserved. This is specified in number of days. When the clean-up job is scheduled as a recurring batch job, this setting will act like a continuously moving window thereby, always leaving the history for the specified number of days intact while deleting the rest. The default is 7 days.

- **Number of hours to execute the job** – Depending on the amount of history to be cleaned up, the total execution time for the clean-up job can vary from a few minutes to a few hours. This parameter must be set to the number of hours that the job will execute. After the clean-up job has executed for the specified number of hours, the job will exit and will resume the clean up the next time it is run based on the recurrence schedule.

A maximum execution time can be specified by setting a max limit on the number of hours the job must run using this setting. The clean-up logic goes through one job execution ID at a time in a chronologically arranged sequence, with oldest being first for the cleanup of related execution history. It will stop picking up new execution ID’s for cleanup when the remaining execution duration is within the last 10% of the specified duration. In some cases, it will be expected that the clean-up job will continue beyond the specified max time. This will largely depend on the number of records to be deleted for the current execution ID that was started before the 10% threshold was reached. The cleanup that was started must be completed to ensure data integrity, which means that cleanup will continue despite exceeding the specified limit. When this is complete, new execution ID’s are not picked up and the clean-up job completes. The remaining execution history that was not cleaned up due to lack of enough execution
Job history clean up and archival

The job history clean up and archival functionality replaces the previous versions of the clean-up functionality. This section will explain these new capabilities.

One of the main changes to the cleanup functionality is the use of the system batch job for cleaning up the history. The use of the system batch job allows Finance and Operations apps to have the clean-up batch job automatically scheduled and running as soon as the system is ready. It is no longer required to schedule the batch job manually. In this default execution mode, the batch job will execute every hour starting at midnight and will retain the execution history for the most recent 7 days. The purged history is archived for future retrieval. Starting with version 10.0.20, this feature is always on.

The second change in the clean-up process is the archival of the purged execution history. The clean-up job will archive the deleted records to the blob storage that DIXF uses for regular integrations. The archived file will be in the DIXF package format and will be available for 7 days in the blob during which time it can be downloaded. The default longevity of 7 days for the archived file can be changed to a maximum of 90 days in the parameters.

Changing the default settings

This functionality is currently in preview and must be explicitly turned on by enabling the flight DMFEnableExecutionHistoryCleanupSystemJob. The staging clean up feature must also be turned on in feature management.

To change the default setting for the longevity of the archived file, go to the data management workspace and select Job history cleanup. Set Days to retain package in blob to a value between 7 and 90 (inclusive). This will take effect on the archives that are created after this change was made.

Downloading the archived package

This functionality is currently in preview and must be explicitly turned on by enabling the flight DMFEnableExecutionHistoryCleanupSystemJob. The staging clean up feature must also be turned on in feature management.

To download the archived execution history, go to the data management workspace and select Job history cleanup. Select Package backup history to open the history form. This form shows the list of all archived packages. An archive can be selected and downloaded by selecting Download package. The downloaded package will be in the DIXF package format and contain the following files:

- The entity staging table file
- DMFDEFINITIONGROUPEXECUTION
- DMFDEFINITIONGROUPEXECUTIONHISTORY
- DMFEXECUTION
- DMFSTAGINGEXECUTIONERRORS

NOTE

If records in the staging tables are not cleaned up completely, ensure that the cleanup job is scheduled to run in recurrence. As explained above, in any clean up execution the job will only clean up as many execution ID's as is possible within the provided maximum hours. In order to continue cleanup of any remaining staging records, the job must be scheduled to run periodically.
It's important to set the time zone for your import job to Coordinated Universal Time (UTC). You might see unexpected dates and times in your imported data if you use a different setting. Without the correct setting, the import process converts the UTC date to the local format, and then system settings converts it again. This dual conversion causes dates to change between applications. For example, the dual conversion could cause an employee's start date to be different between Dynamics 365 Human Resources and Dynamics 365 Finance due to differences in local time zones. Setting the import job to UTC resolves this problem.

1. In Dynamics 365 Finance and Operations, select Data management.
2. Select Import projects, and then select the project.
3. Under Source date format, select CSV-Unicode.
4. Change Timezone to Coordinated Universal Timezone, and change Language to En-US.
This topic provides tips for importing data into the General journal by using the General journal entity.

You can use the General journal entity to import vouchers that have an account or offset account type of Ledger, Customer, Vendor, or Bank. The voucher can be entered as one line, using both the Account field and the Offset account field, or as a multi-line voucher, where only the Account field is used (and the Offset account is left blank on each line). The General journal entity doesn’t support every account type. Instead, other entities exist for scenarios where different combinations of account types are required. For example, to import a project transaction, use the Project expense journal entity. Each entity is designed to support specific scenarios. This means additional fields may be available in entities for those scenarios. However, additional fields might not be available in entities for different scenarios.

Setup

Before you import by using the General journal entity, validate the following setup:

- **Number sequence setup for the journal batch number** – By default, when you import by using the General journal entity, the journal batch number uses the number sequence that is defined in the General ledger parameters. If you set the number sequence for the journal batch number to Manual, a default number isn’t applied. This setup isn’t supported.

- **Financial dimension configuration** – Every organization must define the order of financial dimensions when entities are used to import transactions. The order is defined for the Ledger dimensions integration format, at General ledger > Chart of accounts > Dimensions > Financial dimension configuration for integrating applications > Select data entities. The segments of the ledger account that is imported must have the same order. Otherwise, an error will occur during import.

General journal entity setup

Two settings in Data management affect how the default journal batch number or voucher number is applied:

- **Set-based processing** (on the data entity)
- **Auto-generated** (on the field mapping)

The following sections describe the effect of these settings. They also explain how the system generates batch numbers for journals and voucher numbers.

**Journal batch number**

- The Set-based processing setting on the General journal entity doesn’t affect the way that journal batch numbers are generated.

- If the Journal batch number field is set to Auto-generated, a new journal batch number is created for every line that is imported. This behavior isn’t recommended. The Auto-generated setting is found on the import project, under View map, on the Mapping details tab.

- If the Journal batch number field isn’t set to Auto-generated, the journal batch number is created as follows:
  - If the journal batch number that is defined in the imported file matches an existing, unposted daily
If the journal batch number that is defined in the imported file doesn't match an existing, unposted daily journal, all lines that have the same journal batch number are grouped under a new journal. For example, all lines that have a journal batch number of 1 are imported into a new journal, and all lines that have a journal batch number of 2 are imported into a second new journal. The journal batch number is created by using the number sequence that is defined in the General ledger parameters.

**Voucher number**

- When you use the **Set-based processing** setting on the General journal entity, the voucher number must be provided in the imported file. Every transaction in the General journal is assigned the voucher number that is provided in the imported file, even if the voucher isn’t balanced. Note the following points if you want to use set-based processing, but you also want to use the number sequence that is defined for voucher numbers.
  
  - To enable this functionality, on the journal name that is used for imports, set **Number allocation at posting** to **Yes**.
  
  - A voucher number must still be defined in the imported file. However, this number is temporary and will be overwritten by the voucher number when the journal is posted. Be sure that the lines of the journal are grouped correctly by temporary voucher number. For example, during posting, three lines are found that have a temporary voucher number of 1. The temporary voucher number of all three lines is overwritten by the next number in the number sequence. If those three lines aren’t a balanced entry, the voucher won’t be posted. Next, if lines are found that have a temporary voucher number of 2, this number is overwritten by the next voucher number in the sequence, and so on.

- When you don’t use the **Set-based processing** setting, you do not need to provide a voucher number in the imported file. The voucher numbers are created during import, based on the setup of the journal name (**One voucher only**, **In connection of balance**, and so on). For example, if the journal name is defined as **In connection of balance**, the first line receives a new default voucher number. The system then evaluates the line to determine whether the debits equal the credits. If an offset account exists on the line, the next line that is imported receives a new voucher number. If no offset account exists, the system evaluates whether the debits equal the credits as each new line is imported.

- If the **Voucher number** field is set to **Auto-generated**, the import won’t succeed. The **Auto-generated** setting for the **Voucher number** field isn’t supported.

By default, the General journal entity uses set-based processing. After you evaluate the business requirements for your organization, you can change the **Set-based processing** setting by clicking **Data entities** in the **Data management** workspace. Set-based processing is used to speed up the import process. If you don’t use set-based processing, import of the General journal entity import will be slower.
Data migration is a key success factor in almost every implementation. A primary concern of some customers is the speed that data can be migrated at, especially if there are vast amounts of data and a small cutover window. The Data migration framework is also used to move data as part of business requirements and operations.

The information in this topic represents a set of steps and actions that you can use to optimize the performance of data migration.

NOTE

Testing results in a Tier-1 environment should not be compared or extrapolated to performance in a Tier-2 or higher sandbox environment.

Not all standard entities have been optimized for data migration. Some entities have been optimized for integration with Open Data Protocol (OData), and if a required entity can't be optimized to meet the performance requirements, we recommend that you create a new optimized entity. A developer can accelerate this process by duplicating an existing entity.

Begin the optimization phase by using a subset of the data. For example, if you must import one million records, consider starting with 1,000 records, then increase the number to 10,000 records, and then increase it to 100,000 records.

After you've identified the entities that you will use, you should go through the following sections to explore opportunities for optimization.

Disable change tracking

You can enable and disable change tracking from the list of entities.

1. In the Data management workspace, select the Data entities tile.
2. On the Target entities page, select the entity in the grid, and then, on the Action Pane, on the Change tracking tab, select Disable Change Tracking.

Enable set-based processing

Follow these steps to verify that an entity supports set-based processing.

1. In the Data management workspace, select the Data entities tile.
2. On the Target entities page, find the entity in the grid, and review the value in the Set-based processing column.

For an example that shows how set-based processing can be enabled for the General Journal entity, see Best practices for importing vouchers by using the General journal entity. Not all entities support set-based processing. For example, if you try to enable support set-based processing for the Customer definitions (CustCustomerBaseEntity) entity and save your change, you will receive the following warning message:

Set operations not supported for ‘Customer definitions’ entity.
Here are some important considerations if you must create an entity that allows for set-based processing:

- The data sources can't be read-only.
- The `ValidTimeStateEnabled` property of the data entity view must be set to No.
- All tables on the data sources must have `TableType` property set to Regular.
- The property `QueryType` on the used `Query` can't be set to Union.
- The main data source can't prevent data from being saved across companies. However, embedded data sources allow it.
- The main data source can't prevent data from being saved across partitions. However, embedded data sources allow it.

**Create a data migration batch group**

During cutover, run the data migration while there is little or no other activity. It can be helpful to configure and use a batch group where most or all compute nodes are assigned.

You can configure a batch group on the **Batch group** page (`System administration > Setup > Batch group`).

**Enable priority-based batch scheduling**

In Platform update 31, the new priority-based batch scheduling feature optimizes how batch jobs are run. If contention is identified in the batch framework, consider enabling priority-based batch scheduling.

**Configure the maximum number of batch threads**

To better use parallelism and multithreading, you can configure the maximum number of batch threads per instance of Application Object Server (AOS) by setting the `Maximum batch threads` field on the **Server configuration** page (`System administration > Setup > Server configuration`). Be careful about changing the value of this field. A value that is too high can have negative performance implications. Currently, the default value is 8. You can increase the value to 12 or 16. However, you should not set the field to a value that is more than 16 unless you do significant performance testing.

**Import in batch mode**

Whenever you run an import job, make sure that it's run in batch mode. Otherwise, a single thread will be used to run the job. In this case, the system won't be able to take full advantage of these optimization configurations.

**Clean staging tables**

We recommend that you clean up the staging tables. You can achieve this optimization by scheduling the **Job history cleanup job**. To schedule this job, select the **Job history cleanup** tile in the **Data management** workspace.

**NOTE**

You must first turn on the **Execution history cleanup** feature in the **Feature management** workspace.

**Update statistics**

Before you run a data migration job that involves a large volume of data, consider updating the statistics across the associated tables. This optimization applies specifically to sandbox environments, because it's handled automatically in production environments. You can update statistics for a specific table from LCS. Alternatively, in
Clean the data

The time that is spent on validations and error reporting will increase the total time of the migration. Consider this fact when you import a high volume of invalid or inconsistent data. We recommend that you try to fix and reduce errors that are related to data quality. In this way, you help prevent unnecessary executions of validation and error handling.

Configurations to test during test runs of data migration

The following configurations can affect performance. Therefore, we recommend that you test changes by using different values that are suitable for your scenario.

Configure entity execution parameters

Follow these steps to change the execution parameters for all entities or specific entities.

1. In the Data management workspace, select the Framework parameters tile.
2. On the Data import/export framework parameters page, on the Entity settings tab, select Configure entity execution parameters.
3. On the Entity import execution parameters page, set the Import threshold record count and Import task count fields as appropriate for the desired entities.

Import threshold record count

This value determines the number of records to be processed per thread. By modifying the Import threshold record count you can control how you want to split the import into smaller tasks.

Import task count

This field defines the number of threads that will be used for the data migration job for a specific entity. For example, the Maximum batch threads field for each server is set to 8, and four servers are assigned to the data migration batch group. In this case, the total maximum value of the Import task count field is 32 (= 8 × 4).

If a data entity doesn't support multithreading, you receive an error message when you try to configure the entity. Here is an example:

Custom sequence is defined for the entity ‘Customers V3’, more than one task is not supported.

Validations

Validation logic for record insertions or updates might have been incorporated into the system, or there might be validation of individual fields. If the data migration is mature enough, the time that is spent on imports can be significantly reduced by disabling this validation, if it can be disabled.

Follow these steps to change the settings for each entity.

1. In the Data management workspace, select the Data entities tile.
2. On the Target entities page, select the entity in the grid, and then, on the Action Pane, select Entity structure.
3. On the Entity structure page, set the Run business validations and Run business logic in insert or update method fields as appropriate.

Run business validations

If this check box is selected, the system will run any logic that is written into the validateWrite() method on the table. It will also run any related event handlers.

Run business logic in insert or update method
If this check box is selected, the system will run any logic that is written into the `insert()` or `update()` method on the table. It will also run any related event handlers.

**Call the validateField method on the target**

Follow these steps to run field validation.

1. In the **Data management** workspace, select the **Data entities** tile.
2. On the **Target entities** page, select the entity in the grid, and then, on the Action Pane, select **Modify target mapping**.
3. On the **Map staging to target** page, select the **Call validate Field method** check box for the field that validation should be run for. The `validateField(FieldId p1)` method will then be called for that field.

**Recommendations for optimizing data migration performance**

Here are some general recommendations about the approach that you should use to optimize the performance of data migration:

- Break up large files into smaller chunks. This approach gives the SQL optimizer time to determine whether a new query plan will be optimal.
- Test performance in an appropriate Tier-2 or higher environment.
- Test performance in a mock cutover before go-live.

Testing of data migration performance is an iterative process. We recommend that you collect and compare information about each test to determine the optimal configuration for a specific entity. You should collect and verify some of the following settings:

- The batch group that is used
- The number of batch servers that are assigned to each batch group
- The maximum number of batch threads per batch server

Here is an example of the information that you might collect for each entity.

<table>
<thead>
<tr>
<th>INFORMATION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity</td>
<td>The name of the entity that is being tested</td>
</tr>
<tr>
<td>Number of records</td>
<td>The number of records that are being imported</td>
</tr>
<tr>
<td>Source format</td>
<td>The source format of the data that is being imported</td>
</tr>
<tr>
<td>Change tracking disabled</td>
<td>Yes/No</td>
</tr>
<tr>
<td>Set-based processing</td>
<td>Yes/No</td>
</tr>
<tr>
<td>Import threshold record count</td>
<td>The number of records</td>
</tr>
<tr>
<td>Import task count</td>
<td>The number of tasks</td>
</tr>
<tr>
<td>Run business validations</td>
<td>Yes/No</td>
</tr>
<tr>
<td>Run business logic in insert or update method</td>
<td>Yes/No</td>
</tr>
<tr>
<td>Call validate Field method</td>
<td>Yes/No (potential field list)</td>
</tr>
<tr>
<td>INFORMATION</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Required performance</td>
<td>The amount of time that the import must be completed within to achieve the cutover window</td>
</tr>
<tr>
<td>Actual performance</td>
<td>The actual amount of time that was required to import records</td>
</tr>
</tbody>
</table>

There are more areas where performance can be optimized. For example, you can analyze the specific queries and query plans. However, those processes are covered in other topics.

For more information about performance troubleshooting and optimization, see the following topics:

- Performance troubleshooting using tools in Lifecycle Services (LCS)
- Query cookbook
This article describes design principles for data entities. It also includes guidelines for the names of data entities, fields, relation roles, roles, and OData EntityTypes and EntitySets.

Entity design principles

A data entity should provide a holistic object that encapsulates the relevant business logic in a single consumable contract. The contract is then exposed through application interfaces (APIs), such as OData, import and export, integration, and the programming model. Each data entity should be designed to meet the following goals.

Encapsulation

- Each entity should provide an abstraction between the physical data model and the consumer of the entity. The entity should encapsulate the underlying tables that, together, can define an object that has a specific purpose in the business. Consumers of the entity might be form clients, services, and integration.
- Each entity should encapsulate multiple related tables to represent the domain object. In some situations, single table entities are acceptable.

A single public contract

- The public contract for an entity should be the same across all integration endpoints. For example, the customer entity must expose the same fields or API to both OData and import/export. This principle guarantees that the published schema for an entity is consistent, regardless of the mechanism for consumer interaction.
- When an entity is consumed, the business logic that is executed within the entity during CRUD operations must not vary based on the type of consumer.

Simplicity

- The consumer of an entity should be able to interact with the entity based on the accepted industry or domain definitions of the entity. The behavior details of the entity should be kept hidden and should be prevented from distorting the interaction.
- The consumer of an entity must be able to interact with the entity by using the natural key of the entity. The consumer must not be required to use any keys that are implementation-specific, such as any surrogate key that it generates.

Naming guidelines

Data entity names

<table>
<thead>
<tr>
<th>DO</th>
<th>DON’T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prefix the name with standard prefixes, because of the lack of namespaces.</td>
<td>Don’t include underscores in names.</td>
</tr>
<tr>
<td>Add the postfix “Entity” to the name to prevent conflicts with tables and classes that have the same prefix.</td>
<td>Don’t use abbreviations in conceptual names.</td>
</tr>
</tbody>
</table>
Do

Give the entity a conceptual name that is aligned with the name in the en-us UI. For example, the conceptual name of the entity that exposes records from the InventTable table should be named **ReleasedProduct**, so that the full name of the entity will be **EcoResReleasedProductEntity**.

Result: `<prefix>` `<ConceptualName>Entity`

Data entity field names

**DO**

Give the field name a conceptual name that is aligned with the name in the en-us UI. For example, use **ItemNumber** instead of **ItemID** as the field name to align with the UI, where the label is **Item number**.

Add the postfix "ID," "Number," and so on, to the name of a field that is part of foreign keys to prevent collision with the navigation properties. For example, use **WarehouseID** instead of **Warehouse** as a field name, because **Warehouse** is the navigation method to the **Warehouse** entity.

**DON'T**

Don't include prefixes in field names. For example, a field should not be named **InventLocationId**.

Don't include country/region-specific postfixes in field names. For example, a field should not be named **InventoryProfileID_RU**.

Don't include underscores in field names.

Don't use abbreviations in field names.

Data entity relation role names

**DO**

Use the plural form when you name roles that have a cardinality that is higher than 1. For example, the foreign key on Customer to Party should have the role name of Customers, because the cardinality from Party to Customer is 0...N.

Use the singular form when you name roles that have a cardinality of 0 (zero) or 1. For example, the foreign key on Worker to Person should have the role name of Worker, because the cardinality from Person to Worker is 0..1.

Consider adding the role of the relationship as a prefix. For example, to clearly describe the role of the relationship, name a relationship **BuyingLegalEntity** instead of **LegalEntity**.

**DON'T**

Don't include prefixes in relation role names. For example, don't use the name **WMSWarehouseLocation**, even though the referenced entity includes the prefix "WMS."

Don't add the postfix "Entity" to relation role names. For example, don't use the role name **WarehouseEntity** in a relationship, even though the referenced entity includes the name "Entity." Instead, use the name **Warehouse**.

Don't add country/region-specific postfixes to relation role names. For example, don't use the role name **InventoryProfile_RU**, even though the relationship applies only in an RU country/region format.

Don't include underscores in relation role names.

Data entity relation names

Do

- Give the relation name the same name as the related role name, in singular form. For example, the relation that describes the foreign key from Customer to Party should be named **Party**.
**OData EntityType names**

Do

- Give the EntityType a conceptual name. The name should be the same as the conceptual name of the data entity, but without the prefix and without the “Entity” postfix. For example, `EcoResReleasedProductEntity` becomes `ReleasedProduct`.
- Name the EntityType in singular form.

**OData EntitySet (Entity Collection) names**

Do

- Name the EntitySet in plural form. For example, the EntitySet for the `ReleasedProduct` EntityType is `ReleasedProducts`.

**Number of columns in a data entity**

Note that Microsoft Excel based import/export supports a maximum of 255 columns. If it is expected for an entity to be able to export/import more than 255 columns, then a non-Excel format must be planned or the entity should have less than or equal to 255 columns.
This tutorial shows how to build an entity and how to consume some out-of-band (OOB) entities in an integration scenario. You will also preview how these data entities will be consumed in various integrations scenarios, such as data import and export, integration, and OData services.

When you are ready to build your first entity for production, you will need to:

- Create your own package and model. For more information, see Models and packages.
- Create a new project and set the model property to the one that you just created.

### Prerequisites

This tutorial requires that you access an environment by using Remote Desktop, and that you be provisioned as an administrator on the instance.

Throughout this tutorial, baseUrl refers to the base URL of the instance.

- In the cloud environment, the base URL is obtained from Microsoft Dynamics Lifecycle Services (LCS).
- On a local virtual machine (VM), the base URL is `https://usnconeboxax1aos.cloud.onebox.dynamics.com`.
- Download FMLab sample code to C:. For details, see FMLab sample code.

### Key concepts

- Developing a data entity in Microsoft Visual Studio
- Enabling a data entity for data management and OData services
- Consuming a data entity in integration scenarios

### Business problem

Fleet Management stores customer data in the FMCustomer table and address data in the FMAddressTable table. To access or update customer information, users must access multiple tables. Instead, you can create a business object that functionally represents customer information, and that you can use to build integration solutions.

### Building the FMLabCustomer entity

In this section, you must create a data entity for FMLabCustomer in the Fleet Management model. This entity will be used to manage master data through import/export and integration services. The primary data source is FMCustomer, and the secondary data source is FMAddressTable.

#### Data entity

FMLabCustomerEntity

#### Data entity fields

<table>
<thead>
<tr>
<th>NAME</th>
<th>MAPPING</th>
</tr>
</thead>
<tbody>
<tr>
<td>CellPhone</td>
<td>FMCustomer.CellPhone</td>
</tr>
<tr>
<td>NAME</td>
<td>MAPPING</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>DriverLicense</td>
<td>FMCustomer.DriverLicense</td>
</tr>
<tr>
<td>Email</td>
<td>FMCustomer.Email</td>
</tr>
<tr>
<td>FirstName</td>
<td>FMCustomer.FirstName</td>
</tr>
<tr>
<td>LastName</td>
<td>FMCustomer.LastName</td>
</tr>
<tr>
<td>CustomerGroup</td>
<td>FMCustomer.CustGroup</td>
</tr>
<tr>
<td>AddressLine1</td>
<td>FMAddressTable.AddressLine1</td>
</tr>
<tr>
<td>AddressLine2</td>
<td>FMAddressTable.AddressLine2</td>
</tr>
<tr>
<td>City</td>
<td>FMAddressTable.City</td>
</tr>
<tr>
<td>State</td>
<td>FMAddressTable.State</td>
</tr>
<tr>
<td>ZipCode</td>
<td>FMAddressTable.ZipCode</td>
</tr>
<tr>
<td>Country</td>
<td>FMAddressTable.Country</td>
</tr>
</tbody>
</table>

**Corresponding staging table**

Staging tables are used in import/export scenarios to provide intermediary storage during file parsing and transformation. These tables are also used in connector integration scenarios. In many cases, staging table are mapped 1:1 to an entity. The staging table that corresponds to the `FMLabCustomerEntity` entity is named `FMLabCustomerStaging`.

**Create a new project**

1. In Visual Studio click **File > New > Project**, and then select **Finance and Operations Project**.
2. Right-click the project, click **Properties**, and verify that the project is in the Fleet Management model. If it isn’t, set the **Model** property to **Fleet Management**.

**Add a new data entity to your project**

1. Create a new entity that is named `FMLabCustomerEntity`. Right-click you project, and then click **Add > New item**. The **Add New Item** dialog box opens.
2. Select **Data Entity**, and then set the **Name** property to `FMLabCustomerEntity`.
3. Click **Add**.
4. In the **Data entity** wizard, specify the properties for the data entity that you’re creating. Use the values that are shown in the following image.

**NOTE**

The name of an entity must not have `_` or any numeric digits (0…9). Using these characters may result in mapping errors later.
5. Click **Next**. For more information about the function of each property, see “Categories of entities” and “Building an entity” in Data entities overview.

6. Add fields to the new entity from your data source, as shown in the following image. You can add fields from the primary data source, **FMCustomer**. For this entity, clear the check box for the **Image** and **LicenseImage** container types to simplify testing.

7. Rename the data entity fields to reflect public data contract standards, or click **Convert labels to field names** to generate names from the existing labels.

8. On the line for the **DriverLicense** field, select the **Is mandatory** check box. This field will be used as the natural key for the entity.

9. In the **Data source** field, select **PrimaryAddress**. Notice that the **PrimaryAddress** data source is automatically added because of automatic expansion or the surrogate foreign key replacement of **AddressID**.
10. Select the fields from the **PrimaryAddress** data source that you want to be part of your entity. Additionally, rename the following fields to reflect proper public data contract naming:

- **PrimaryAddress_AddressLine1** > **AddressLine1**
- **PrimaryAddress_AddressLine2** > **AddressLine2**
- **PrimaryAddress_City** > **City**
- **PrimaryAddress_State** > **State**
- **PrimaryAddress_ZipCode** > **ZipCode**
- **PrimaryAddress_Country** > **Country**

11. Click **Finish**. A data entity item and staging table are added to the project.

**Build your project**

1. In Solution Explorer, right-click your project, and then click **Properties**.

2. Change the value of the **Synchronize database on build** property to **True**, and then click **OK**. This property must be set only one time per project.

**NOTE**

Entities are created as views in Microsoft SQL Server, and staging tables are also added. Therefore, you must sync a database when you build entities.
3. On the Visual Studio toolbar, click **Build > Build Solution** to build the project.

4. Verify that the build doesn't contain any errors. At this point in the tutorial, warnings are allowed.

**Visually validate and customize an entity**

1. In Solution Explorer, right-click the **FMLabCustomerEntity** node, and then click **Open**. The designer for the entity opens in the middle pane.

2. Validate the properties of the **FMLabCustomerEntity** entity. Select the entity in Solution Explorer, and compare the **Properties** pane values to the following image.

3. Set the **Label** property to **Fleet Lab Customers**.

4. In the left pane, click **Data Sources > FMCustomer > Data Sources > FMAddressTable**.
5. Change the Is Read Only property to No. This is a known issue. Eventually, the value will be set to Yes or No automatically, based on the type of join. The value should be Yes for composition scenarios, and No for associations (surrogate foreign key expansions). This property enables the data source to be read/write.

![Image of data source properties]

6. In the FMLabCustomerEntity designer, click Keys > EntityKey, and then expand the Fields node. Verify that the list of fields matches the following image.

![Image of field list]

7. To visually validate the staging table that will be used for import/export, open the FMLabCustomerStaging table in the designer, and then select the FMLabCustomerStaging node.
8. Click **FMLabCustomerStaging > Fields**. In the following image, the standard fields of the staging tables are selected. All entity staging tables have these standard fields. The image also shows the data fields that belong on this data entity.

9. In Solution Explorer, right-click your project, and then select **Rebuild** to rebuild and synchronize the project.

**Testing data entities**

Entities can be tested by using various methods in X++, through data import/export, or through integrations. In this section, we'll explore scenarios for validating entities.
Test the entity by using X++ code

One of the most common ways of interacting with data entities is through X++, by using a unit test or a runnable job to validate that the entities have been built. In this example, we will use a runnable job.

1. In Solution Explorer, click Add > New item > Runnable class to add a runnable class to your project.

2. Copy and paste the following code into the class to test your data entity.

   ```java
   public static void main(Args _args)
   {
       FMLabCustomerEntity customer;
       str license = "License";
       Random r = new Random();
       int rand = r.nextInt();
       license = license + int2str(rand);

       //Create a new record in FM lab customer entity
       customer.clear();
       customer.FirstName = "Bob";
       customer.LastName = "Smith";
       customer.DriverLicense = license;
       customer.insert();

       info(strfmt("Tried to insert customer '%1 %2' with license %3",
                   customer.FirstName, customer.LastName, customer.DriverLicense));

       //Display newly created record
       select forupdate customer where customer.DriverLicense==license;
       info(strfmt("Found newly created customer '%1 %2' with license %3",
                   customer.FirstName, customer.LastName, customer.DriverLicense));

       //Now delete the record from the entity
       customer.delete();
       select customer where customer.DriverLicense==license ;
       info(strfmt("Deleted customer does not exist: license- %1", customer.DriverLicense));
   }
   ```

3. Run the code in debugger to set it as a startup object.

4. To validate the entity, view the Infolog in the debugger window or in notifications on the website. You will see that three successful messages are logged. You will also see the actions that were taken.

Importing data by using entities

Data entities that have the Data Management Enabled property can be used to import and export data in various file formats. In this section, you will import data in a CSV file format for the FMLabCustomer entity.

File import

After you create your data entity, you can validate import/export.

1. Create a sample CSV file that you can import. Copy the following text, and save it as FM Lab Customer Import.csv.
2. Click **User Dashboard > Data management.**

3. In the **Data Management** workspace, click the **Import** tile.

4. On the **Import** page, enter the import details, as shown in the following image.

5. Click the **Upload data** button next to the **Upload file for entity** field, and select the CSV file that you created.

6. After the file is uploaded, you will notice that the entity is added to the middle section. You will also receive an error that states that the mapping isn't valid. A few fields aren't mapped correctly between the source file and the target entity.

7. In the entities list, click **View map**.

8. **AddressLine1** and **AddressLine2** are two fields in the source that aren't mapped to target fields. In the visual mapper, or details view, map these fields as follows, and then click **Save**:

   - AddressLine1 – Address1
   - AddressLine2 – Address2
Consuming entities by using OData

In this section, you will learn how to expose and consume an entity for OData. Before you begin, verify that the Fleet demo data is loaded from the client: [baseURL]/?f=FMSetup

**Review the FleetRental entity and add a navigation property for OData**

You will review the existing FleetRental entity and then create a relationship from one data entity to another. This relationship will be used as a navigation property for OData entities.

1. In Solution Explorer, verify that you're in the FMEntityLab project.

2. In Application Explorer, search for FMEntityLab, right-click it, and then select Add to Project.
3. In Application Explorer, search for **FMCustomerEntity**, right-click it, and then select **Add to project**.

4. In Solution Explorer, right-click **FMRentalEntity**, and then select **Open**.

5. In the view designer, select the root node, **FMRentalEntity**, and review the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsPublic</td>
<td>Yes</td>
<td>When this property is set to <strong>Yes</strong>, the entity is visible by using the OData application programming interface (API).</td>
</tr>
<tr>
<td>Public Entity Name</td>
<td>FleetRental</td>
<td>The name that will be used in the OData APIs for <strong>EntityType</strong>.</td>
</tr>
<tr>
<td>Public Collection Name</td>
<td>FleetRentals</td>
<td>The name that will be used for the OData collection entity.</td>
</tr>
</tbody>
</table>

6. In the view designer, expand the **Relations** node.

7. Select **Customer Relation**, and then click **Delete**.

8. Right-click **Relations**, and then select **New > Relation**.

9. Select **Relation1**, and set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardinality</td>
<td>ZeroMore</td>
</tr>
<tr>
<td>Name</td>
<td>CustomerRelation</td>
</tr>
<tr>
<td>Related Data Entity</td>
<td>FMCustomerEntity</td>
</tr>
<tr>
<td>Related Data Entity Cardinality</td>
<td>ExactlyOne</td>
</tr>
<tr>
<td>Related Data Entity Role</td>
<td>CustomerRole</td>
</tr>
<tr>
<td>Relationship Type</td>
<td>Association</td>
</tr>
<tr>
<td>Role</td>
<td>Rental</td>
</tr>
</tbody>
</table>

10. In the view designer, right-click the **CustomerRelation** node, and then select **New > Normal**.

11. Right-click the new node under **CustomerRelation**, and then select **Properties**.

12. Set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>CustomerDriverLicenseThis is the foreign key field on FMRentalEntity.</td>
</tr>
<tr>
<td>Related Field</td>
<td>DriverLicenseThis is the unique key on FMCustomerEntity.</td>
</tr>
</tbody>
</table>
The following image shows the relation in Visual Studio.

![Visual Studio screen showing relation between FMRentalEntity and FMCustomerEntity]

13. On the **BUILD** menu, click **Build Solution** to save your changes and build the project. You can view the build progress in the **Output** window.

14. To update the OData endpoint with the changes, you must run an `iisreset` command. Open a **Command Prompt** window as an administrator, and enter `iisreset`.

You've now created a navigation property between **FMRentalEntity** and **FMCustomerEntity**.

**Use standard OData syntax to retrieve data**

In this section, you will use some of the standard OData syntax to navigate and query the OData entities that are exposed in the Fleet Management model. First, follow these steps to enable Internet Explorer to view JSON formatted data.

1. Close all Internet Explorer windows.
2. Go to C:\FMLab, and select and double-click the `json-ie.reg` file.
3. In the **Registry Editor** dialog box, click **Yes**.
4. Click **OK**.

You can now use Internet Explorer to explore some OData URLs.

1. Start Internet Explorer, and enter the following URL in the address bar: `[baseURL]/data/$metadata` You will see all the metadata that is associated with OData entities.

   **NOTE**
   The metadata might take a few minutes to appear the first time that you access it. In the XML, you can see all of the properties and navigation properties associated with the OData entities.

2. In the browser, find **FleetRental**. The following image shows the metadata of the **FleetRental** entity, together with the new relationship, **NavigationProperty**.
3. To view all the customers in the Fleet Management application in JSON format, enter the following URL into the address bar of your browser: `[baseURL]/data/FleetCustomer`

**NOTE**
Entity names are case-sensitive.

4. If you don't want to retrieve all properties for the customers, you can retrieve just selected properties. For example, to retrieve only `FirstName` and `LastName`, enter the following URL: `[baseURL]/data/FleetCustomers?$filter=FirstName.Lastname`

5. You can also apply filters. For example, to filter on all customers where `FirstName=Phil`, enter the following URL: `[baseUrl]/data/FleetCustomers?$filter=FirstName%20eq%20'Phil'`

**NOTE**
These URLs won't work if you copy and paste them. You must manually enter them in the address bar.

6. To retrieve all the `Rental` records, together with all details of the customers, enter the following URL: `[baseURL]/data/FleetRentals?$expand=CustomerRole` The following example shows a `Rental` record, together with details of the linked customer, in JSON format.
Add an action to OData entity

Actions provide a way to inject behaviors into the data model. In Dynamics 'AX 7,' you add actions by adding a method to the data entity and then decorating the method with specific attributes. In this section, we'll walk through the steps for adding an action.

1. In Solution Explorer, right-click `FMRentalEntity`, and then select View code.

2. Copy the following code lines, and paste them into the Code window.

```csharp
public class FMRentalEntity extends common
{
    [SysODataActionAttribute("ReturnRental", true)]
    public str ReturnRental()
    {
        //do something
        return "Rental was successfully returned. Thanks for your business";
    }
}
```

3. On the BUILD menu, click Rebuild Solution to save your changes and build the project. You can view the build progress in the Output window.

4. To update the OData endpoint with the changes, you must run an `iisreset` command. Open a Command Prompt window as an administrator, and enter `iisreset`.

The action that you just added can be invoked through code, as you will see in the next section.

Consume the OData API from an external console application

In this section, you will use a console application to consume the OData endpoints that are exposed in the Fleet Management application. The console application first creates a new customer and then creates a new reservation for that customer. This tutorial shows how easy it is to use OData together with standard .NET Windows Communication Foundation (WCF) data service libraries to integrate with Dynamics AX.


2. On the File menu, click Open > Project/Solution.
3. In the Open Project dialog box, browse to C:\FMLab\Odata4ConsoleApplication, and then select Odata4ConsoleApplication.csproj.

4. Click Open. The Odata4ConsoleApplication project appears in Solution Explorer.

5. In Solution Explorer, double-click OdataProxyGenerator.tt.

6. In the code editor, replace the following string with your organization's URL.

```
<baseURL> public const string MetadataDocumentUri = "<baseURL>/data/"
```

7. Save the OdataProxyGenerator.tt file.

8. In the Save of Read-only file dialog box, click Overwrite. The proxy class for the OData metadata endpoint is generated. This operation might take a few minutes.


10. Replace the value of the dynamicsBaseUri variable with your organization's URL.

11. Verify that there is a final closing slash (/) in the URL, and then click Save.

12. In the Save of Read-only file dialog box, click Overwrite.

13. Press F5 to run the application, and then follow the instructions in the output console window. The application might prompt you for your Dynamics AX credentials. After the application has run, the new customer and the corresponding reservation are created.

14. Follow these steps to verify that the new reservation appears on the Rental page:

   a. Start Internet Explorer, and enter the following URL in the address bar: [baseURL]?mi=FMRental The FMRental page shows the list of rentals.

   b. At the bottom of the list, click Next to view the next page. On this page, you can see that the reservation was created for the new customer that you added.
This completes the walkthrough, where you’ve seen an external client interacting with the Fleet Management model by using OData endpoint.

**Casing rules in data entities**

**XML format**
During an export, the entity name and the field names are exported in uppercase. If there is a need to apply a transformation, the transformation must use uppercase in all references.

During an import, data management accepts input file in any casing. However, care must be taken to have the same format for a given attribute/element in the file. When applying a transformation, ensure that the transformation is using the same casing rules in all references as in the incoming file.

**Excel format**
During an export, column names will be exported in uppercase. Imports are not case sensitive.

**CSV format**
During an export, column names will be exported in uppercase. Imports are not case sensitive.

**Tips and tricks**

**Max join limits**
During entity development, ensure that the overall structure of the entity does not exceed the max join limit of 26. This is the default limit. Increasing the join limit is not recommended because it can have unintended consequences. If this limit is exceeded, the entity will most likely fail to process records and will result in the following SQL error. We also recommend managing the total number of columns in the entity to avoid this error.

```
Cannot create a row of size xxx which is greater than the allowable maximum row size of 8060
```

**Exporting container fields**
If an entity has container fields and these fields need to be exported, the entity must implement `getFieldsToBeConvertedToFile` to allow each container field to export its data value to a separate file. This allows for container fields to be exportable and at the same time, prevents making the entity export file (core entity data without the container fields) unreadable. If `getFieldsToBeConvertedToFile` is not implemented, then these fields will not be exported but the rest of the entity data will export as usual.

**Additional resources**

[Develop entities for data migration](#)
Every data entity has properties that let you override the same property values on the tables or views that are the data sources of that entity. Your choices affect the behavior of the entity. In the following table, the first column lists the properties that are discussed in this topic. The top row lists the levels where the property is found in the entity designer. The levels are listed in order of increasing granularity: the data source level is more granular than the entity level but less granular than the field level.

<table>
<thead>
<tr>
<th>Property</th>
<th>Entity Level</th>
<th>Data Source Level</th>
<th>Field Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadOnly</td>
<td>Applies</td>
<td>Applies</td>
<td>.</td>
</tr>
<tr>
<td>AllowEdit</td>
<td>.</td>
<td>.</td>
<td>Applies</td>
</tr>
<tr>
<td>AllowEditOnCreate</td>
<td>.</td>
<td>.</td>
<td>Applies</td>
</tr>
<tr>
<td>Mandatory</td>
<td>.</td>
<td>.</td>
<td>Applies</td>
</tr>
</tbody>
</table>

**Entity level**

In the designer for your data entity, when you click the name at the root node, the **Properties** pane includes the **Is Read Only** property. The following table describes the behavioral differences between the **Yes** and **No** values of this property.

<table>
<thead>
<tr>
<th>GROUP</th>
<th>PROPERTY NAME</th>
<th>DISPLAY NAME</th>
<th>VALUES</th>
<th>DEFAULT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>GROUP</th>
<th>PROPERTY NAME</th>
<th>DISPLAY NAME</th>
<th>VALUES</th>
<th>DEFAULT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| Behavior    | IsReadOnly    | Is Read Only | No, Yes | No      | • **No**: Data modification operations (CUD) are allowed, unless an individual data source node in the entity’s designer is set to `IsReadOnly = Yes`.  

• **Yes**: Only read operations are allowed, regardless of the `IsReadOnly` settings on the individual data source nodes in the entity’s designer. |

You would set **IsReadOnly** to **Yes** for entities that are consumed mainly for export.

### Data source level

If a data entity has three data sources, you might want to allow processes to use the entity to modify the data in one of the data sources but not in the other two. A read-only data source can be used for lookup purposes. You can use the entity designer to achieve this extra degree of granular control. Under the entity’s **Metadata > Data Sources** node, you can select an entity node and then set the **IsReadOnly** property value for that one data source. The following table describes the interaction between the **IsReadOnly** settings at the data source level and the entity level.
### Field level

At the field level, the `AllowEdit` and `AllowEditOnCreate` properties are available instead of an `IsReadOnly` property. The two `Allow` properties include `Auto` as a third available value. The `Auto` value inherits the value that is on the field in the underlying table.

#### NOTE

The `Auto` value isn't available for unmapped fields, such as computed or virtual fields.
This behavior is enforced for all consumers – X++, OData, and so on.

The No and Yes values do not override the setting on the field in the underlying table.

| No: | Users aren't allowed to modify the data for this field in a new record. |
| Yes: | Users are allowed to modify the data for this field for a new record. |

This behavior is enforced for all consumers – X++, OData, and so on.

The No and Yes values do not override the setting on the field in the underlying table.

<table>
<thead>
<tr>
<th>GROUP</th>
<th>PROPERTY NAME</th>
<th>DISPLAY NAME</th>
<th>VALUE</th>
<th>DEFAULT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>isn't available for unmapped fields, such as computed or virtual fields.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GROUP</th>
<th>PROPERTY NAME</th>
<th>DISPLAY NAME</th>
<th>VALUE</th>
<th>DEFAULT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GROUP</td>
<td>PROPERTY NAME</td>
<td>DISPLAY NAME</td>
<td>VALUE</td>
<td>DEFAULT</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------</td>
<td>---------------</td>
<td>--------------</td>
<td>------------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>Behavior</td>
<td>AllowEdit</td>
<td>Allow edit</td>
<td>Auto, No, Yes</td>
<td>Auto</td>
<td>The behavior is the same as the behavior for AllowEditOnCreate, but it applies to updates to existing records instead of new records that are being created. This behavior is enforced for all consumers – X++, OData, and so on.</td>
</tr>
<tr>
<td>Behavior</td>
<td>Mandatory</td>
<td>Mandatory</td>
<td>Auto, No, Yes</td>
<td>Auto</td>
<td><strong>Auto</strong>: The property is inherited from the underlying table field. This behavior is enforced for all consumers – X++, OData, and so on.</td>
</tr>
</tbody>
</table>

[IMPORTANT] The **No** and **Yes** values do not override the setting on the field in the underlying table.
This article provides information about computed and virtual fields, which are the two types of unmapped fields that a data entity can have. The article includes information about the properties of unmapped fields, and examples that show how to create, use, and test them.

The sample code is targeted towards creating or modifying an entity that is a part of solution that you own. Extending an existing entity requires slight modifications.

### Overview

A data entity can have additional *unmapped* fields beyond those that are directly mapped to fields of the data sources. There are mechanisms for generating values for unmapped fields:

- Custom X++ code
- SQL executed by Microsoft SQL Server

The two types of unmapped fields are computed and virtual. Unmapped fields always support read actions, but the feature specification might not require any development effort to support write actions.

#### Computed field

- Value is generated by an SQL view computed column.
- During read, data is computed by SQL and is fetched directly from the view.
- For writes, custom X++ code must parse the input value and then write the parsed values to the regular fields of the data entity. The values are stored in the regular fields of the data sources of the entity.
- Computed fields are used mostly for reads.
- If possible, it's a good idea to use computed columns instead of virtual fields, because they are computed at the SQL Server level, whereas, virtual fields are computed row by row in X++.

#### Virtual field

- Is a non-persisted field.
- Is controlled by custom X++ code.
- Read and write happens through custom X++ code.
- Virtual fields are typically used for intake values that are calculated by using X++ code and can’t be replaced by computed columns.
- Search and filtering on virtual fields are not supported.

#### Properties of unmapped fields

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>NAME</th>
<th>TYPE</th>
<th>DEFAULT VALUE</th>
<th>BEHAVIOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATEGORY</td>
<td>NAME</td>
<td>TYPE</td>
<td>DEFAULT VALUE</td>
<td>BEHAVIOR</td>
</tr>
</tbody>
</table>
### Data

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>NAME</th>
<th>TYPE</th>
<th>DEFAULT VALUE</th>
<th>BEHAVIOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>IsComputedField</td>
<td>NoYes</td>
<td>Yes</td>
<td>• Yes – The field is synchronized as a SQL view computed column. Requires an X++ method to compute the SQL definition string for the column. The virtual column definition is static and is used when the entity is synchronized. After that, the X++ method is not called at run time. • No – The field is a true virtual field, where inbound and outbound values are fully controlled through custom code.</td>
</tr>
<tr>
<td>Data</td>
<td>ComputedFieldMethod</td>
<td>String</td>
<td></td>
<td>A static <code>DataEntity</code> method in X++ to build the SQL expression that will generate the field definition. This property is disabled and irrelevant if the property <code>IsComputedField</code> is set to <code>No</code>. The method is required if the property <code>IsComputedField</code> is set to <code>Yes</code>.</td>
</tr>
<tr>
<td>Data</td>
<td>ExtendedDataType</td>
<td>String</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Example: Create a computed field

In this example, you add a computed field to the `FMCustomerEntity` entity. For reads, the field combines the name and address of the customer into a nice format. For writes, your X++ code parses the combined value into
its separate name and address values, and then the code updates the regular name and address fields.

1. In Microsoft Visual Studio, right-click your project, and add the existing **FMCustomerEntity**.

2. In Solution Explorer, right-click the **FMCustomerEntity** node, and then click **Open**.

3. In the designer for **FMCustomerEntity**, right-click the **Fields** node, and then click **New > String Unmapped Field**.

4. Rename the new field **NameAndAddress**.

5. Update properties of the **NameAndAddress** unmapped field, as shown in the following screenshot.

6. Go to **FMCustomerEntity > Methods**. Right-click the **Methods** node, and then click **New**. Ensure that the method name matches the **DataEntityView Method** property value of the unmapped computed field.

7. Paste the following X++ code into the method. The method returns the combined and formatted **NameAndAddress** value.

```
NOTE
The server keyword is required.
```
private static server str formatNameAndAddress() // X++
{
    DataEntityName   dataEntityName= tablestr(FMCustomerEntity);
    List             fieldlist = new list(types::String);
    //Format name and address to look like following
    /////John Smith, 123 Main St, Redmond, WA 98052
    fieldlist.addEnd(SysComputedColumn::returnField(dataEntityName, identifierstr(FMCustomer),
        fieldstr(FMCustomer, FirstName)));
    fieldlist.addEnd(SysComputedColumn::returnLiteral(" "));
    fieldlist.addEnd(SysComputedColumn::returnField(dataEntityName, identifierstr(FMCustomer),
        fieldstr(FMCustomer, LastName)));
    fieldlist.addEnd(SysComputedColumn::returnLiteral(";
    fieldlist.addEnd(SysComputedColumn::returnField(dataEntityName, identifierstr(FMAddressTable),
        fieldstr(FMAddressTable, AddressLine1)));
    fieldlist.addEnd(SysComputedColumn::returnLiteral(", ");
    fieldlist.addEnd(SysComputedColumn::returnField(dataEntityName, identifierstr(FMAddressTable),
        fieldstr(FMAddressTable, City)));
    fieldlist.addEnd(SysComputedColumn::returnLiteral(", ");
    fieldlist.addEnd(SysComputedColumn::returnField(dataEntityName, identifierstr(FMAddressTable),
        fieldstr(FMAddressTable, State)));
    fieldlist.addEnd(SysComputedColumn::cast(
        SysComputedColumn::returnField(dataEntityName, identifierstr(FMAddressTable),
            fieldstr(FMAddressTable, ZipCode)), "NVARCHAR"));
    return SysComputedColumn::addList(fieldlist);
}

T-SQL for the computed column.

( Cast (( ( T1.firstname ) + ( N' ' ) + ( T1.lastname ) + ( N'; ' ) +
    ( T5.addressline1 )
    + ( N' ' ) + ( T5.city ) + ( N' ' ) + ( T5.state ) + (N'
    )
    ) +
    ( Cast(T5.zipcode AS NVARCHAR) ) ) AS NVARCHAR(100))
    AS
    NAMEANDADDRESS

TIP
If you receive error in data entity synchronization because of computed columns, it's easier to come up with the
SQL definition in Microsoft SQL Server Management Studio (SSMS) before using it in X++.

8. Rebuild the project.
9. Synchronize the database. Don't forget this step. You can do this by going to Dynamics 365 >
    Synchronize database > Synchronize.

Example: Create a virtual field

In this example, you add a virtual field to the FMCustomerEntity entity. This field displays the full name as a
combination of the last name and first name. X++ code generates the combined value.

1. In the designer for the FMCustomerEntity entity, right-click the Fields node, and then click New >
    String Unmapped Field.
2. In the properties pane for the unmapped field, set the Name property to FullName.
3. Set the Is Computed Field property to No. Notice that you leave the DataEntityView Method empty.
4. In the FMCustomerEntity designer, right-click the Methods node, and then click Override > postLoad. Your X++ code in this method will generate the values for the virtual field.

5. Paste the following X++ code in for the postLoad override. Notice that the postLoad method returns void.

```x++
public void postLoad()
{
    super();
    //Populate virtual field once entity has been loaded from database
    //Format full name - "Doe, John"
    this.FullName = this.LastName + ", " + this.FirstName;
}
```

6. Compile your project.

Example: Use a virtual field to receive and parse an inbound field

Imagine that an external system sends the name of a person as a compound value that combines the last and first names in one field that comes into our system. However, our system stores the last and first names separately. For this scenario, you can use the FullName virtual field that you created. In this example, the major addition is an override of the mapEntityToDataSource method. When update is called, mapEntityToDataSource methods are invoked for each data source.

1. In the designer for the FMCustomerEntity, right-click the Methods node, and then click Override > mapEntityToDataSource.

2. Paste the following X++ code in for the mapEntityToDataSource method.
public void mapEntityToDataSource(DataEntityRuntimeContext entityCtx, 
DataEntityDataSourceRuntimeContext dataSourceCtx) 
{
    super(entityCtx, dataSourceCtx);
    // Check if desired data source context is available
    if (dataSourceCtx.name() == "FMCustomer")
    {
        FMCustomer dsCustomer = dataSourceCtx.getBuffer();
        // Find position of "," to parse full name format "Doe, John"
        int commaPosition = strfind(this.FullName, ",",0,strlen(this.FullName));
        // Update FirstName and LastName in the data source buffer to update
        dsCustomer.LastName = substr(this.FullName,0,commaPosition-1);
        dsCustomer.FirstName = substr(this.FullName, commaPosition+1, strlen(this.FullName));
    }
}

Test the computed and virtual fields

The following main method tests your computed and virtual fields. Both fields are tested in a read action, and the virtual field is tested in an update action.

1. For this example, ensure that you have the data set named Fleet Management (migrated). The data set is available from the dashboard in the browser. Click the menu icon in the upper-right corner, click the APP LINKS menu, and then scroll to find the data set named Fleet Management (migrated).

2. Paste the following X++ code into the startup object of your project. Run your project.

```java
public static void main(Args _args) // X++
{
    FMCustomerEntity customer;
    // Using transactions to avoid committing updates to database
ttsbegin;
    // SELECT single customer entity record from the database
    select customer
    where customer.Email == "phil.spencer@adatum.com";
    // Read full name (Virtual Field)
    info(customer.FullName);
    // Read formatted NameAndAddress(computed Field)
    info(customer.NameAndAddress);
    // UPDATE full name (virtual field)
    customer.FullName = "Doe, John";
    customer.update();
    // Reselect data from database to get updated information
    select customer
    where customer.Email == "phil.spencer@adatum.com";
    // Read full name (virtual field)
    info(customer.FullName);
    ttsabort;
}
```

Note on computed column generation failures

If the X++ method that generates the SQL for a computed column throws an exception, DbSync catches the exception, sets the value of that column to NULL, and logs a warning.

Developers are advised to check configuration keys manually in computed column methods to avoid hitting a NULL value, if the generation failed.
This topic provides information about how data entities interact with the cross-company concept. To understand this aspect of data entities, you must understand how tables and views apply the cross-company concept. Therefore, this topic begins with a brief review of tables and views, and then explains how data entities are related.

Review of tables and views for cross-company

Each table has a **SaveDataPerCompany** property, and each view has a **AllowCrossCompany** property. The following table describes these two properties.

<table>
<thead>
<tr>
<th>TABLE</th>
<th>VIEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property name</td>
<td>SaveDataPerCompany</td>
</tr>
<tr>
<td>Relevant CRUD mode</td>
<td>CUD</td>
</tr>
<tr>
<td>Timing of effect</td>
<td>Run time, Design time</td>
</tr>
<tr>
<td>Meaning of value = Yes</td>
<td>At design time, the system automatically adds a <strong>dataAreaId</strong> field to the table, even though the field isn’t displayed in the Application Object Tree (AOT). Every record in the table is tagged with the company (or legal entity) that it belongs to. The system automatically adds a filter to the SQL <strong>Where</strong> clause to limit the returned set of rows to one <strong>dataAreaId</strong> value.</td>
</tr>
<tr>
<td>Meaning of value = No</td>
<td>The system does not add a <strong>dataAreaId</strong> field to the table. The table is said to be a shared table, because none of its records contain any formal company-specific data.</td>
</tr>
</tbody>
</table>

Comparisons within **AllowCrossCompany** = No

In the following screenshot, the **CustomerList** view has two data sources:

- **Root** – CustTable, which has its **SaveDataPerCompany** property set to **Yes**.
- **Non-root** – DirPartyTable, which has its **SaveDataPerCompany** property set to **No**.
CREATE VIEW [dbo].[CUSTOMERLIST] AS
SELECT T1.accountnum AS ACCOUNTNUM,
T1.dataareaid AS DATAAREAID, -- AllowCrossCompany =No caused this line.
T1.partition  AS PARTITION,
T1.recid      AS RECID,
T2.partition  AS PARTITION#2,
T2.name       AS NAME
FROM custable T1
CROSS JOIN dirpartytable T2
WHERE ( T1.party = T2.recid
  AND ( T1.partition = T2.partition ) )
CREATE VIEW [dbo].[CUSTOMERLISTPARTY] 
AS 
SELECT T1.name AS NAME, 
T1.partition AS PARTITION, 
T1.recid AS RECID, 
T2.partition AS PARTITION#2, 
T2.accountnum AS ACCOUNTNUM 
FROM dirpartytable T1 
CROSS JOIN custtable T2 
WHERE ( T2.party = T1.recid 
AND ( T2.partition = T1.partition ) ) 
go

By swapping the positions of the two data source tables in the CustomerList view, you make the DirPartyTable table the root data source.

In this case, the SQL Create View statement is the same, except for the following two differences:

- The FROM clause mentions DirPartyTable first and CustTable second.
- The SELECT column list does not include the line for dataAreaId (because DirPartyTable has its SaveDataPerCompany property set to No.)

Limitations of tables and views

In some cases, the cross-company control features of tables and views aren’t as granular control as you might require. Here are the limitations:

- Company or legal entity fields other than the system dataAreaId field can’t be recognized or treated automatically in the that way dataAreaId can.
- The cross-company behavior for views is too restricted to the properties of the root data source, even when non-root data sources have a dataAreaId field.

For example, this might happen if the legal entity information is in LegalEntityRecId, or if shared tables don’t have a dataAreaId column.

Design time: Setting the PrimaryCompanyContext property
Data entities help you overcome the limitations of tables and view where cross-company functionality is concerned. Data entities have a **PrimaryCompanyContext** property, where you can specify the entity field to use for company identification. This property provides flexibility and granular control in the following ways:

- The field can be from any data source of the entity and isn't limited to fields of the root data source.
- The field can be any field that is extended from the **DataAreaId** extended data type (EDT), and isn't limited to an underlying system **dataAreaId** field.
- You can use the **PrimaryCompanyContext** property even when the entity has only shared tables as its data sources, if this makes sense for your specific situation.

The following screenshot shows the value set for the **PrimaryCompanyContext** property on the **FMCustGroupEntity** entity.

![Screenshot showing the value set for the PrimaryCompanyContext property on the FMCustGroupEntity entity.]

When the **PrimaryCompanyContext** value is set to a non-empty value, the entity can't behave as a shared entity. The **dataAreaId** field is added to the SQL `CREATE View` statement.

```sql
CREATE VIEW [dbo].[FMCUSTGROUPENTITY]
AS
SELECT T1.custgroup AS GROUPNAME,
      T1.description AS DESCRIPTION,
      T1.dataareaid AS DATAAREAID,  -- dataAreaId is added.
      T1.recversion AS RECVERSION,
      T1.partition AS PARTITION,
      T1.recid AS RECID
FROM fmcustgroup T1
```

**Run time: The behavior of data entities for cross company**

In the context of X++ code, the cross-company behavior of data entities resembles the behavior of tables. If the **PrimaryCompanyContext** property for an entity has no value and is empty, the entity behaves like a shared table.

**X++ when PrimaryCompanyContext is set**

The following table describes the behavior of a data entity under CRUD access when the **PrimaryCompanyContext** property is set to a field value. Both X++ and OData accesses are described.

<table>
<thead>
<tr>
<th></th>
<th>X++</th>
<th>ODATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read  (R)</td>
<td>By default, results are always filtered by <strong>dataAreaId = current company</strong>; and cross-company data can be fetched by the &lt;code&gt;cross company&lt;/code&gt; option.</td>
<td>Results are not filtered by <strong>dataAreaId</strong>. The consumer must filter explicitly.</td>
</tr>
<tr>
<td><strong>X++</strong></td>
<td><strong>ODATA</strong></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>Write (CUD)</td>
<td>CUD access to the data entity always occurs in the context of the current company. If cross-company CUD access to the entity is required, use the <code>changeCompany</code> keyword.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CUD access to the entity can be accomplished by the consumer for any company by setting the value of the <code>PrimaryCompanyContext(myDataAreaId)</code> field. The framework handles the necessary <code>ChangeCompany</code> action.</td>
<td></td>
</tr>
</tbody>
</table>

The following X++ code example accesses **FMCustGroupEntity**, which has its `PrimaryCompanyContext` property set to `dataAreaId`. |
X++ when PrimaryCompanyContext is empty

When the `PrimaryCompanyContext` property is set on the data entity, a `dataAreaId` field is created in the view schema and mapped to the `PrimaryCompanyContext` field. The following table describes the behavior of a data entity under CRUD access when the `PrimaryCompanyContext` property is empty. Both X++ and OData accesses are described.

<table>
<thead>
<tr>
<th>X++</th>
<th>ODATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><strong>X++</strong></td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Read (R)</strong></td>
<td>Results aren’t filtered, because no system <code>dataAreaId</code> field is created on the view schema.</td>
</tr>
<tr>
<td><strong>Write (CUD)</strong></td>
<td>There is no primary company context to set. Therefore, CUD access to the entity is always in the context of the current company.</td>
</tr>
</tbody>
</table>

In the current example, the `FMCustomerGroupGlobalEntity` entity has no value assigned to its `PrimaryCompanyContext` property.

> ![Diagram of FMCustomerGroupGlobalEntity entity](image)

However, a `dataAreaId` field from the FMCustGroup table is mapped to the `FMCustomerGroupGlobalEntity` entity as a regular field that is named `LegalEntity`. In this example, the FMCustGroup table is the root data source for `FMCustomerGroupGlobalEntity`. However, we are using this `dataAreaId` field in an informal way that bypasses the automatic mechanisms of the system. All these details are shown in the following screenshot of the `LegalEntity` field.

> ![Diagram of LegalEntity field](image)

**NOTE**

Although the terms *legal entity* and *data entity* both use the word *entity*, don’t confuse them. Legal entities and data entities are two entirely different concepts. When the `PrimaryCompanyContext` property is empty, the SQL `Create View` statement usually contains no mention of a system `dataAreaId` column. However, in the current example, `dataAreaId` is “half-mentioned” because of the `LegalEntity` regular field on the data entity. This field is shown in the following SQL statement.
CREATE VIEW [dbo].[FMCUSTOMERGROUPGLOBALENTITY] AS

SELECT T1.custgroup AS NAME,
       T1.description AS DESCRIPTION,
       T1.dataareaid AS LEGALENTITY, -- dataareaid is named LegalEntity.
       T1.recversion AS RECVERSION,
       T1.partition AS PARTITION,
       T1.recid AS RECID
FROM   fmcustgroup T1

Purpose of this example

This example has two purposes:

- Show shared data by default, even though the backing table might be company-specific.
- Enable the consumer of the data entity to filter on or apply dataAreaid if this is required, by using the regular field that is named LegalEntity.

Test data

The following screenshot of the Table browser page shows the test data that is in the FMCustomerGroupGlobalEntity entity before the X++ test code is run.

X++ code

Here's how the X++ test code works with the shared entity:

- It accesses the data entity in shared mode for reads.
- It accesses the data entity with one specific company when a new record is created.
class GlobalCrossCompanyXPlusPlusTest
{
    /// <summary>
    /// Runs the class with the specified arguments.
    /// </summary>
    /// <param name="_args">The specified arguments.</param>
    public static void main(Args _args)
    {
        FMCustomerGroupGlobalEntity customerGroup;

        // Reads
        // Returns record(s) for global entity from all legal entities
        while select Name,LegalEntity from customerGroup
        {
            info(strfmt("%1-%2",customerGroup.Name,customerGroup.LegalEntity));
        }

        // Writes
        // Create record in company based on legal entity
        // In this case, even without ChangeCompany, data source insert would happen
        // In company identified by legal entity automatically. However, there is a
        // Known bug around this area.
        changecompany("ceu")
        {
            customerGroup.clear();
            customerGroup.Name = "CEU test-1";
            customerGroup.LegalEntity = "CEU";
            customerGroup.insert();
        }

        select firstonly Name,LegalEntity from customerGroup
        where customerGroup.Name == "CEU test-1";
        info(strfmt("%1-%2",customerGroup.Name,customerGroup.LegalEntity));
    }
}
This article provides scenarios that are applicable from an implementation perspective for both configuration keys and country/region.

**Customer table schema**

<table>
<thead>
<tr>
<th>FIELD NAME</th>
<th>FIELD LABEL</th>
<th>COUNTRY CONTEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustNum</td>
<td>Customer number</td>
<td></td>
</tr>
<tr>
<td>CustName</td>
<td>Customer name</td>
<td></td>
</tr>
<tr>
<td>EinvoiceEANNum</td>
<td>EAN</td>
<td>DK</td>
</tr>
<tr>
<td>FiscalCode</td>
<td>Fiscal code</td>
<td>IT</td>
</tr>
</tbody>
</table>

**Sample data**

<table>
<thead>
<tr>
<th>CUSTNUM</th>
<th>CUSTNAME</th>
<th>EINVOICEEANUM(DK)</th>
<th>FISCALCODE(IT)</th>
<th>DATAAREAID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Contoso Denmark</td>
<td>AA</td>
<td>(Empty)</td>
<td>DK</td>
</tr>
<tr>
<td>2</td>
<td>Contoso Italy</td>
<td>(Empty)</td>
<td>DD</td>
<td>IT</td>
</tr>
</tbody>
</table>

**Sample entity**

<table>
<thead>
<tr>
<th>FIELD NAME</th>
<th>COUNTRY CONTEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustomerNumber</td>
<td></td>
</tr>
<tr>
<td>CustomerName</td>
<td></td>
</tr>
<tr>
<td>EAN</td>
<td>DK</td>
</tr>
<tr>
<td>FiscalCode</td>
<td>IT</td>
</tr>
</tbody>
</table>

**Scenario – Field level only**

Isaac, a developer, builds a customer entity that has fields that contain regional settings. The entity is consumed through OData.

**For read operations:** The consumer of the entity uses this information to complete an effective regional mapping. The consumer ignores the fields that aren't required for that region. For example, consumers in Denmark (DK) are concerned with reading the values of the EAN field and core fields only.

**For write operations:** The consumer of the entity uses this information to identify only the fields that are required to populate data. The consumer expects validation to occur for regional fields and associated core fields.

**Behaviors – Fields only**
<table>
<thead>
<tr>
<th>SCENARIO</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td>Entities automatically inherit localization properties from underlying fields.</td>
</tr>
<tr>
<td>Design</td>
<td>Developers can’t override or set localization properties on entity fields. These properties should be inherited only from tables. Only override on unmapped fields.</td>
</tr>
<tr>
<td>Read behavior (OData metadata)</td>
<td>The consumer of an entity from OData will have metadata or annotations to specify which fields are localized.</td>
</tr>
<tr>
<td>Read behavior (Data management)</td>
<td>In import/export fields, metadata displays country/region values, so that this information is obvious to the end user.</td>
</tr>
<tr>
<td>Read behavior</td>
<td>During cross-company read operations, data from localized fields is displayed only if the context matches. Note that this should already be implemented through the table/view.</td>
</tr>
<tr>
<td>Read behavior (Performance)</td>
<td>During company-specific read operations, localized fields are dropped from the query when the context doesn’t match.</td>
</tr>
<tr>
<td>Write</td>
<td>During write operations to localized fields, hard errors occur if the fields don’t match the context.</td>
</tr>
<tr>
<td>(Shared table)</td>
<td>If the data source or fields contain a country/region that is a shared (global) table, all operations are ignored, just as if no keys are applied.</td>
</tr>
</tbody>
</table>

**Behavior – Data source**

The behavior of configuration keys and a country/region that are applied at the data source resembles the behavior of fields. These values are inferred from the data source, just as if they are applied to the field level. Here’s an example.

```
Entity E1
   |_ Data Source 1 (DS1)
       | Field1
       | Field2
   |_ Data Source 2 (DS2) UK
       | Field3
       | Field4
```

**Evaluation at the entity E1 level**
Entity E1
| _F1
| _F2
| _F3 (UK inferred)
| _F4 (UK inferred)
Describes support for inheritance patterns in data entities.

Patterns

There are several ways to create entities for tables that involve inheritance:

- **Leaf/concrete type as data source**: If a concrete type is used as a data source, fields are displayed for both the base type and the current type. For example, in the following screen shots, if DirPerson is the data source, data source fields from both DirPerson and DirPartytable appear.
Abstract type/non-leaf as data source: If a non-leaf type is used as a data source, fields are displayed for both the base type and the current type, but fields from any derived types aren't displayed. Fields from derived types must be added from derived data sources, as shown in the following screen shot.
Data Entity View wizard

You can use the Data Entity View wizard to create data entities where the primary data source (and additional data sources) can be tables that are involved in inheritance.

**NOTE**

Currently, the wizard doesn't support derived data sources. It shows only fields from the current type or the base type. After you create an entity, you can manually modify it to display derived data sources.

The following screen shots show a data entity that was created by using the wizard, where DirPartyTable is the primary data source.

1. Update the data source table to DirPartyTable.
2. Update the data source table to **DirPartyTable**.

---

**Run time**

There is run-time behavior for entities that related to inheritance.

**Creating entities for specified types**

In this example, we create separate **Person** and **Organization** entities. The primary data source for the **Person** entity is DirPerson, and the primary data source for the **Organization** entity is DirOrganization. This approach, which is reflected in the following screen shots, doesn’t require that you write any special run-time code.
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data entity field name</th>
<th>Data type</th>
<th>E1 Type name</th>
<th>Is Mandatory</th>
<th>Label</th>
<th>Height/Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResidenceType</td>
<td>ResidenceType</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MatTitle</td>
<td>MatTitle</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>InTitle</td>
<td>InTitle</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ParentSuffix</td>
<td>ParentSuffix</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ParentTfn</td>
<td>ParentTfn</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PhoneticName</td>
<td>PhoneticName</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PhoneticInName</td>
<td>PhoneticInName</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NameSequence</td>
<td>NameSequence</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CommunicationSystem</td>
<td>CommunicationSystem</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Creating entities for generalized types

In this example, we create a single entity, Party, that can be used for both Person and Organization. The primary data source is DirPartyTable, and derived data sources are DirPerson and DirOrganization. The new entity contains the following kinds of fields:

- **Common attributes** – Attributes that aren't specific to Person or Organization, such as Name. These fields are mapped to DirPartyTable.
- **Person-specific attributes** – Gender, Marital Status, and so on. These fields are mapped to derived data source DirPartyTable_DirPerson.
- **Organization-specific attributes** – OrgNumber, ABC, and so on. These fields are mapped to derived
Mapping fields from base and multiple derived types in a single data entity is a design-time task. However, at run time, we must specify when each derived type should be created. This can be based on fields such as `InstanceRelationType`, or a computed column can be created to use `String` to represent different types. In the `Party` entity example, a `PartyType` computed column can be created to represent the `Person` and `Organization` derived types. The following code snippet illustrates this approach.

```java
private static server str PartyType()
{
    str type = CASE + SysComputedColumn::returnField(tablestr(PartyEntity),
        tablestr(DirPartyTable), fieldstr(DirPartyTable, InstanceRelationType)) +
        ' WHEN ' + int2str(tablenum(DirOrganization)) + ' THEN ' Organization' +
        ' WHEN ' + int2str(tablenum(DirPerson)) + ' THEN ' Person' +
    " ELSE 'Party' " +
    ' END ';
    return type;
}
```

In this example, the `Party` type is computed by using the `InstanceRelationType` column on `DirPartyTable`. This approach works for reading data. However, to do Create or Update operations, you must write code where you override the `initializeEntityDataSource` method on the data entity, based on type, and set a correct instance of the derived type for the data source run-time context buffer.
public void initializeEntityDataSource(DataEntityRuntimeContext entityCtx, 
DataEntityDataSourceRuntimeContext dataSourceCtx) 
{
    DirPerson   person;
    DirOrganization org;
    DirPartyTable party;

    if (dataSourceCtx.name() == tables[DirPartyTable] &&
        (dataSourceCtx.getDatabaseOperation() == DataEntityDatabaseOperation::Insert ||
         dataSourceCtx.getDatabaseOperation() == DataEntityDatabaseOperation::Update)
    )
    {
        if (dataSourceCtx.getDatabaseOperation() == DataEntityDatabaseOperation::Update)
        {
            party = dataSourceCtx.getBuffer();
            if (this.PartyType == 'Person')
            {
                person = party.as DirPerson;
                dataSourceCtx.setBuffer(person);
            }
            else
            {
                org = party.as DirOrganization;
                dataSourceCtx.setBuffer(org);
            }
        }
        else
        {
            if (this.PartyType == 'Person')
            {
                dataSourceCtx.setBuffer(new DictTable(tableNum(DirPerson)).makeRecord());
            }
            else
            {
                dataSourceCtx.setBuffer(new DictTable(tableNum(DirOrganization)).makeRecord());
            }
        }
    }
    super(entityCtx, dataSourceCtx);
}
This topic provides information about the natural key expansion of surrogate foreign key fields and the expansion of child/parent relations.

**Natural key expansion of surrogate foreign keys**

A surrogate foreign key field’s extended data type must be `RefRecId` or a derivative. The natural key expansion of a surrogate foreign key field uses the rules in the following list. These rules are listed in the order of evaluation.

1. **Replacement key** – The replacement key fields
2. **Primary key** – The primary index key fields
3. **Alternate key** – The first unique alternate key
4. **Auto-identification key** – The auto-identification fields

Surrogate foreign key fields that are nested in the natural key are recursively expanded. Recurring nested surrogates are limited to the first occurrence. If you select the `is mapped` property of a surrogate foreign key (that is, if you set the property to `true`), the related data source is automatically added to the entity, and the `is mapped` property of each field in the related data source’s natural key is selected. In addition, any nested surrogate foreign key data sources are recursively added to the entity. If you clear the `is mapped` property of a surrogate foreign key (that is, if you set the property to `false`), the related data source are automatically removed and unmapped from the entity and any nested surrogate foreign key data sources. The effect of selecting and clearing the `is mapped` property of a surrogate foreign key field differs from the effect of using the `Add data source` and `Remove data source` buttons. If you add a surrogate foreign key data source, the `is mapped` property of the parent data source surrogate foreign key field isn’t automatically set to `true`. If you are removing a surrogate foreign key data source, the `is mapped` property of the parent data source surrogate foreign key field isn’t automatically set to `false`. By default, the `is mapped` property of the root data source’s surrogate foreign key field is set to `true`. Therefore, by default, surrogate foreign key relations are expanded to one level.

**Expansion of parent/child relations**

Parent/child relations are composition/extension relations that are stored as a relation on the child table. The parent table doesn’t detect the relation. A parent/child relation can be either a surrogate foreign key relation or a natural foreign key. Parent/child relations use the following rules:

- The collection of child tables is obtained by querying the cross-reference database for “type reference” relations to the parent table.
- The child table must belong to the same table group as the parent table.
- The relation between the child and parent `relationship type` property must be `association`, `composition`, `link`, or `aggregation`.
- The relation between the child and parent `cardinality` property must be `exactly one` or `zero or one`.
- The relation between the child and parent `related table cardinality` property must be `exactly one` or `zero or one`.

By default, parent/child relations aren’t expanded. You must add them by using the `Add data source` button. To change the default behavior for a project in Microsoft Visual Studio, set the `Expand parent/child` option to...
true.

Using label text as field names

The following rules enable label text to be used as the field name when the option is selected (true):

- All whitespace is removed from the label text.
- The label text is truncated to 77 characters, and then a unique three-digit numeric identifier (000 through 999) is added.
- The label text is passed to the `NamedElement.IsValid` method. If the name is valid, it can be used as the field name. Otherwise, the label text isn’t used, and the field name remains unchanged.

The Is mandatory field

The default value of data entity fields is auto. This value is used unless it’s explicitly changed. For relations, the related field’s Is mandatory property is set based on the value of the field’s Is mandatory value.

- If the field’s Is mandatory property and the related field’s Is mandatory are both true, both fields are explicitly set to true.
- If the field’s Is mandatory property and the related field’s Is mandatory are both false, both fields are explicitly set to false.

If the field’s Is mandatory property and the related field’s Is mandatory property differ in value, both fields remain unchanged. In this case, the default value of auto is used.
Every data entity has properties that let you override the same property values on the tables or views that are the data sources of that entity. Your choices affect the behavior of the entity. In the following table, the first column lists the properties that are discussed in this topic. The top row lists the levels where the property is found in the entity designer. The levels are listed in order of increasing granularity: the data source level is more granular than the entity level but less granular than the field level.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>ENTITY LEVEL</th>
<th>DATA SOURCE LEVEL</th>
<th>FIELD LEVEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadOnly</td>
<td>Applies</td>
<td>Applies</td>
<td></td>
</tr>
<tr>
<td>AllowEdit</td>
<td></td>
<td></td>
<td>Applies</td>
</tr>
<tr>
<td>AllowEditOnCreate</td>
<td></td>
<td></td>
<td>Applies</td>
</tr>
<tr>
<td>Mandatory</td>
<td></td>
<td></td>
<td>Applies</td>
</tr>
</tbody>
</table>

### Entity level

In the designer for your data entity, when you click the name at the root node, the Properties pane includes the **Is Read Only** property. The following table describes the behavioral differences between the **Yes** and **No** values of this property.

<table>
<thead>
<tr>
<th>GROUP</th>
<th>PROPERTY NAME</th>
<th>DISPLAY NAME</th>
<th>VALUES</th>
<th>DEFAULT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>

You would set IsReadOnly to Yes for entities that are consumed mainly for export.

Data source level

If a data entity has three data sources, you might want to allow processes to use the entity to modify the data in one of the data sources but not in the other two. A read-only data source can be used for lookup purposes. You can use the entity designer to achieve this extra degree of granular control. Under the entity’s Metadata > Data Sources node, you can select an entity node and then set the IsReadOnly property value for that one data source. The following table describes the interaction between the IsReadOnly settings at the data source level and the entity level.

<table>
<thead>
<tr>
<th>GROUP</th>
<th>PROPERTY NAME</th>
<th>DISPLAY NAME</th>
<th>VALUES</th>
<th>DEFAULT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| Behavior | IsReadOnly | Is Read Only | No, Yes | No | • No: Data modification operation(s) (CUD) are allowed, unless an individual data source node in the entity's designer is set to IsReadOnly = Yes.
• Yes: Only read operations are allowed, regardless of the IsReadOnly settings on the individual data source nodes in the entity's designer. |
<table>
<thead>
<tr>
<th>GROUP</th>
<th>PROPERTY NAME</th>
<th>DISPLAY NAME</th>
<th>VALUES</th>
<th>DEFAULT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behavior</td>
<td>IsReadOnly</td>
<td>Is Read Only</td>
<td>No, Yes</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

- **No**: Data modification operations (CUD) are allowed on the data source, unless IsReadOnly is set to Yes at the entity level.
- **Yes**: Only operations are allowed, regardless of the IsReadOnly setting on the entity.

### Field level

At the field level, the AllowEdit and AllowEditOnCreate properties are available instead of an IsReadOnly property. The two Allow properties include Auto as a third available value. The Auto value inherits the value that is on the field in the underlying table.

**NOTE**
The Auto value isn't available for unmapped fields, such as computed or virtual fields.

<table>
<thead>
<tr>
<th>GROUP</th>
<th>PROPERTY NAME</th>
<th>DISPLAY NAME</th>
<th>VALUE</th>
<th>DEFAULT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behavior</td>
<td>AllowEditOnCreate</td>
<td>Allow edit on create</td>
<td>Auto, No, Yes</td>
<td>Auto</td>
<td></td>
</tr>
</tbody>
</table>

- **Auto**: The property is inherited from the underlyin g table field.

![NOTE]

The Auto value
This behavior is enforced for all consumers – X++, OData, and so on.

<table>
<thead>
<tr>
<th>GROUP</th>
<th>PROPERTY NAME</th>
<th>DISPLAY NAME</th>
<th>VALUE</th>
<th>DEFAULT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>isn’t available for unmapped fields, such as computed or virtual fields.</td>
</tr>
</tbody>
</table>

- **No:**
  Users aren’t allowed to modify the data for this field in a new record.

- **Yes:**
  Users are allowed to modify the data for this field for a new record.

This behavior is enforced for all consumers – X++, OData, and so on.

![IMPORTANT](1)

The **No** and **Yes** values do *not* override the setting on the field in the underlying table.
<table>
<thead>
<tr>
<th>GROUP</th>
<th>PROPERTY NAME</th>
<th>DISPLAY NAME</th>
<th>VALUE</th>
<th>DEFAULT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Behavior</td>
<td>AllowEdit</td>
<td>Allow edit</td>
<td>Auto, No, Yes</td>
<td>Auto</td>
<td>The behavior is the same as the behavior for AllowEditOnCreate, but it applies to updates to existing records instead of new records that are being created. This behavior is enforced for all consumers – X++, OData, and so on.</td>
</tr>
<tr>
<td>Behavior</td>
<td>Mandatory</td>
<td>Mandatory</td>
<td>Auto, No, Yes</td>
<td>Auto</td>
<td><strong>Auto</strong>: The property is inherited from the underlying table field. This behavior is enforced for all consumers – X++, OData, and so on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[IMPORTANT] The No and Yes values do &lt;em&gt;not&lt;/em&gt; override the setting on the field in the underlying table.</td>
</tr>
</tbody>
</table>
This topic describes how data entity values are validated, how default values can be provided, and how to use fields that are not mapped to data source values, but instead contain virtual or computed data (unmapped fields).

**Validations**

Validations can be defined on the tables that back up entities, at both the field level and the record level. Validations can also be defined at the data entity level.

**Table (data source) vs. entity validation**

Entities are backed by tables (data sources), and validations are defined for these tables at both the field level (`Table.validateField()`) and the record level (`Table.validateWrite()`). The validations are respected by data entities that are built by using those tables. Although these validations are intrinsic to the tables that back a data entity, validations can also be defined at the data entity level. Like table-based validations, entity-based validations can be written at the field level (`DataEntity.validateField()`) or the record level (`DataEntity.validateWrite()`).

**Table-based validation behavior**

Table validations are fired automatically as a part of the CUD operations. `Table.ValidateField`, `AllowEdit`, `AllowEditOnCreate` Field-level validations are fired automatically when you perform inserts or updates on the data entity. This is true for all paths (X++, OData, and so on). These validations occur during the mapping process, when fields are mapped from an entity to individual data sources.

![Diagram](image)

After the field values from the data entity are copied to mapped data source fields, field validations are run on the set fields. Validations include table-level `validateField`, which validates `AllowEdit` and `AllowEditOnCreate`. If a validation fails because of an error, validation for the remaining fields continues. Finally, validation checks whether any error occurred during the validation process for any of the data sources. If there was an error, the process errors out at this point, and table-level `validateWrite()` isn't called. To skip
validateField for a back-end table, a consumer can call `DataEntity.skipDataSourceValidateField(Int _DataEntityFieldId, Boolean _skip)`. Note that the field ID for this method is the field ID of the data-entity mapped field, not the back-end table field. By using the following API, you can skip validation for a particular field, regardless of the consumer.

```java
public void partialEntity(DataEntityRuntimeContext entityCtx)
{
    this.skipDataSourceValidateField(_fieldId, true)
    super(entityCtx);
}
```

**Table.ValidateWrite** Record-level **ValidateWrite** validations that are defined in back-end tables are fired automatically when you perform data-entity inserts and updates. This is true for all paths (X++, OData, and so on). These validations occur just before the actual insert or update is applied to the data source. If the validation fails, an error is thrown, and the process stops for other data sources.

To skip **validateWrite** for all back-end tables for a data entity, a consumer can call `DataEntity.skipDataSourceValidateWrite(Boolean _skip)`. This method turns **validateWrite** on or off for all data sources. By using the following API, you can skip validation for a particular data source, regardless of the consumer.
Table.ValidateDelete Record-level ValidateDelete validations that are defined in back-end tables are fired automatically when you perform data entity deletes. This is true for all paths (X++, OData, and so on). These validations occur just before the delete is applied to the data source. If the validation fails, an error is thrown, and the process stops for other data sources.

To skip validateDelete for all back-end tables for a data entity, a consumer can call DataEntity.skipDataSourceValidateDelete(Boolean _skip). This method turns validateDelete on or off for all data sources. By using the following API, you can skip validation for a particular data source, regardless of the consumer.
Entity-based validation behavior

<table>
<thead>
<tr>
<th>VALIDATION</th>
<th>TARGET</th>
<th>CALLER</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataEntity.ValidateField</td>
<td>• Data types</td>
<td>• Is called automatically from OData</td>
</tr>
<tr>
<td></td>
<td>• Mandatory relationships (both tables and extended data types [EDTs])</td>
<td>• Is called by the form engine when a field is modified</td>
</tr>
<tr>
<td></td>
<td>• Any custom validation</td>
<td>• Isn’t called automatically if an insert/update is fired from X++ code</td>
</tr>
<tr>
<td></td>
<td>• Doesn’t call validateField for underlying mapped table fields</td>
<td></td>
</tr>
<tr>
<td>DataEntity.ValidateWrite</td>
<td>• Mandatory columns</td>
<td>• Is called automatically from OData</td>
</tr>
<tr>
<td></td>
<td>• Relationships (both tables and EDTs)</td>
<td>• Is called by the form engine when a record is saved</td>
</tr>
<tr>
<td></td>
<td>• Any custom validation</td>
<td>• Isn’t called automatically if an insert/update is fired from X++ code</td>
</tr>
<tr>
<td></td>
<td>• Doesn’t call table-level validateWrite for underlying tables</td>
<td></td>
</tr>
<tr>
<td>DataEntity.ValidateDelete</td>
<td>• DeleteActions</td>
<td>• Is called automatically from OData</td>
</tr>
<tr>
<td></td>
<td>• Any custom validation</td>
<td>• Is called by the form engine when a record is deleted</td>
</tr>
<tr>
<td></td>
<td>• Doesn’t call table-level validateDelete for underlying tables</td>
<td>• Isn’t called automatically if a delete is fired from X++ code</td>
</tr>
</tbody>
</table>

Defaults

Default values can be provided for initializations and rows.

Initializations

DataEntity.initValue: A data entity is initialized with default values and by using any custom logic that is present in entity-level initValue. This method isn’t called automatically when an insert or update is performed on a data entity from X++. It must be called explicitly if it’s required. The method is called automatically by the form engine when a new record is created. DataEntity.initValue doesn’t call the initValue method for back-end tables that are used in the data entity. Table.initValue: Table-level initValue, as defined for back-end tables, is fired when you perform a data entity insert. This is true for all paths (X++, OData, and so on). Table.initValue
is run just before the entity is mapped to data source fields.

To skip entity-level `initValue` for all back-end tables for a data entity, a consumer can call `DataEntity.skipDataSourceInitValue(Boolean _skip)`. This method turns `initValue` on or off for all data sources. By using the following API, you can skip `initValue` for a particular field, regardless of the consumer.

```csharp
/// <summary>
/// </summary>
/// <param name="uvCtx"> </param>
/// <param name="uvdsCtx"> </param>
public void initializeEntityDataSource(DataEntityRuntimeContext uvCtx, DataEntityDataSourceRuntimeContext uvdsCtx)
{
    if (uvdsCtx.name() == tablestr(DirPartyTable)
    & uvdsCtx.getDatabaseOperation() == DataEntityDatabaseOperation::Insert)
    {
        uvdsCtx.skipInitValue(true);
    }
    super(uvCtx, uvdsCtx);
}
```

**DefaultRow**

`DataEntity.DefaultRow: DataEntity.DefaultRow` is used in conjunction with `defaultField` and `getDefaultingDependencies` to provide defaults. It isn’t called automatically by X++ or the form engine.

**Table.DefaultRow**

`Table.DefaultRow: Table.DefaultRow` is called automatically for each data source after mapping is completed, and before the insert and validation on the data source.
Unmapped fields

A data entity can have *unmapped* fields in addition to those fields that are directly mapped to fields of the data sources. There are two mechanisms for generating values for unmapped fields:

- Custom X++ code
- SQL that is run by Microsoft SQL Server

The two types of unmapped fields are *virtual* and *computed*. Unmapped fields always support read actions, but the feature specification might not require any development effort to support write actions.

**Virtual field**

- A non-persisted field.
- Controlled by custom X++ code.
- Read and writes occur through custom X++ code.
- Typically used for intake values that are calculated by using X++ code and can't be replaced by computed columns.

**Computed field**

- The value is generated by an SQL view computed column.
- During reads, data is computed by SQL and fetched directly from the view.
- For writes, custom X++ code must parse the input value and then write the parsed values to the regular fields of the data entity. The values are stored in the regular fields of the data sources of the entity.
- Used mostly for reads.
- It's a good idea to use computed columns instead of virtual fields whenever you can, because computed columns are computed at the SQL Server level, whereas virtual fields are computed row by row in X++.

Properties of unmapped fields
<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>NAME</th>
<th>TYPE</th>
<th>DEFAULT VALUE</th>
<th>BEHAVIOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>IsComputedField</td>
<td>NoYes</td>
<td>Yes</td>
<td>• Yes: The field is synchronized as a SQL view computed column. An X++ method is required to compute the SQL definition string for the column. The virtual column definition is static and is used when the entity is synchronized. After that, the X++ method isn't called at run time. • No: The field is a true virtual field, where inbound and outbound values are fully controlled through custom code.</td>
</tr>
<tr>
<td>Data</td>
<td>ComputedFieldMethod</td>
<td>String</td>
<td></td>
<td>A static DataEntity method in X++ is used to build the SQL expression that generates the field definition. This property is disabled and irrelevant if the IsComputedField property is set to No. The method is required if the IsComputedField property is set to Yes.</td>
</tr>
<tr>
<td>Data</td>
<td>ExtendedDataType</td>
<td>String</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Unmapped field comparison**

<table>
<thead>
<tr>
<th>VIRTUAL FIELD</th>
<th>COMPUTED FIELD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Metadata properties | Is computed = No | • Is Computed = Yes  
• Computed Field Method = static method |
|---------------------|------------------|--------------------------------------------------|
| Read                | • X++ (override `postLoad`)  
• Row by row | • SQL computed column  
• Set-based read possible |
| Write               | X++ (override `mapEntityToDataSource`) | X++ (override `mapEntityToDataSource`) |
| Advantages          | • Unbound to the schema, keeps the public contract the same, but the implementation can change  
• Call X++ methods | Faster reads, large export can occur directly from the view |

### Examples

The following table provides a computed example if a `UnitOfMeasure` relationship exists, and displays that in an unmapped field.

<table>
<thead>
<tr>
<th>VIRTUAL FIELD</th>
<th>COMPUTED FIELD</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>On postLoad()</code>//Check to see if record exists in UnitOfMeasureInternalCode.UnitOfMeasure//Set hasFixedInternalCode value based on the field (this.UnitOfMeasure)this.HasFixedInternalCodeVirtual = NoYes:Yes; else this.HasFixedInternalCodeVirtual = NoYes:No;</td>
<td><code>On computedFieldMethod()</code>//Desired SQL computed column statement(CASE WHEN T2.RECID IS NULL THEN 0 ELSE 1 END) AS INT</td>
</tr>
</tbody>
</table>
Security and data entities

NOTE
Data entities do not support the Extensible Data Security (XDS) concepts.

Entry points

Data entities support entry point security. This support resembles the support that menu items and pages have. To give you flexibility when you define a security model, data entities allow for a separate security configuration for each integration mode. Currently, two entry points/integration modes are identified for a data entity.

<table>
<thead>
<tr>
<th>ENTRY POINT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data services</td>
<td>The ability to use OData services (API) for the entity.</td>
</tr>
<tr>
<td>Data management</td>
<td>The ability to use asynchronous integration options for the entity, such as import/export and connector integration.</td>
</tr>
</tbody>
</table>

Target scenarios

Data entities can support multiple categories of scenarios. Each category might have to be secured separately.

- **Data management (file-based import/export, and so on)** – Typically, a data manager performs these scenarios. These scenarios might provide access to data that isn’t usually accessible through the UI for the client. Therefore, you will often want to secure data management scenarios independently of access to the related page, so that a data manager can perform only import/export operations.

- **General integration via OData** – Many integration scenarios require that data entities be exposed as services, so that data can be accessed via OData (for example, from an online storefront or a Process Lifetime Management [PLM] system). Often, you will want to control access to data entities that are built for this purpose independently of page access. In other words, you will want to grant access to the service interface without granting access through the client UI.

- **Microsoft Office integration (Edit in Excel, and so on)** – Office integration scenarios also require that data entities be accessed via OData. However, from an end-user perspective, these scenarios can be viewed as a natural extension of the client, where, for example, Microsoft Excel is used to simplify some editing tasks. Therefore, there is usually no reason to secure the Microsoft Office integration independently of page access.

Privilege/duty mapping

Depending on the target scenarios for a data entity, you should create one or more new privileges, and extend existing duties. Alternatively, you can map the new privileges to duties that are created specifically for the target scenario. This approach helps guarantee that no user is granted more access than the user requires for the scenario.

The following table shows the privileges that you should create. It also explains how you should map these privileges to duties. If your data entity is intended to support more than one scenario, you should create the combined set of privileges and duty mappings.
The data entity is intended for data management.

Create the following new privileges:
- `<DataEntity>Import`
  - Grant=Create
  - IntegrationMode=Data Management
- `<DataEntity>Export`
  - Grant=Read
  - IntegrationMode=Data Management

Extend the relevant data management duties with the new privileges. For more information, see the “Data administrator role” section later in this topic.

The data entity is intended for general integration via OData.

Create the following new privileges:
- `<DataEntity>View`
  - Grant=Read
  - IntegrationMode=Data Services
- `<DataEntity>Maintain`
  - Grant=Delete
  - IntegrationMode=Data Services

Create new duties for the integration scenario, and map the relevant new privileges to these duties. For more information, see the “Duty naming guidelines” section later in this topic.

The data entity is intended for Microsoft Office integration.

Create the following new privileges:
- `<DataEntity>View`
  - Grant=Read
  - IntegrationMode=Data Services
- `<DataEntity>Maintain`
  - Grant=Delete
  - IntegrationMode=Data Services

Extend the relevant existing duties that provide access to the related pages with the new privileges.

Because the approach that is described in the preceding table complies with the principle of least privilege, we recommend that you use it. Nevertheless, in some situations, you can use the following simpler approach. However, be aware that this approach might be less secure. It might also be slightly harder to maintain and extend.

The data entity is intended for data management, general integration via OData, and Microsoft Office integration.

Create the following new privileges:
- `<DataEntity>View`
  - Grant=Read
  - IntegrationMode=All
- `<DataEntity>Maintain`
  - Grant=Delete
  - IntegrationMode=All

1. Extend the relevant data management duties with the new privileges.
2. Create new duties for the integration scenario, and map the relevant new privileges to these duties.
3. Extend the relevant existing duties that provide access to the related page with the new privileges.

When you use this approach, a data manager who is granted access to import/export data can also access the system from a web service. Likewise, a user who is granted access to the page that is associated with a data entity can also access the system from a web service. This user will be prevented from data import/export only if the user hasn’t been granted the related data management duty.
Duty naming guidelines

When you create data entities for specific integration scenarios, you should also create separate duties. These duties grant the external application or service the required access to the data entities. The duties that you create should follow the same naming guidelines as the corresponding duties that provide access through the client UI. However, you should add a "using services" suffix.

<table>
<thead>
<tr>
<th>DUTY TYPE</th>
<th>DUTY OBJECT NAME SUFFIX</th>
<th>DUTY NAME TEMPLATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>...IntegrationEnable</td>
<td>Enable ... using services</td>
</tr>
<tr>
<td>Record</td>
<td>...IntegrationMaintain</td>
<td>Maintain ... using services</td>
</tr>
<tr>
<td>Authorize</td>
<td>...IntegrationApprove (Release, Confirm, Journalize)</td>
<td>Approve (Release, Confirm, Journalize) ... using services</td>
</tr>
<tr>
<td>Inquire</td>
<td>...IntegrationInquire</td>
<td>Inquire into ... using services</td>
</tr>
</tbody>
</table>

Here are some examples of duty names that follow these guidelines:

- Maintain route master using service
- Inquire into case progress using services

Data administrator role

The DataManagementApplicationAdministrator security role enables an associated user to have full import/export capabilities in the Data management workspace. This security role has two security duties for each of the five data entity categories that are assigned to it. One duty is for importing data via data entities of the associated category, and one for exporting data via data entities of the associated category. Therefore, a total of 10 security duties are assigned to this security role:

- DataManagementApplicationDocumentEntitiesMaintain
- DataManagementApplicationDocumentEntitiesView
- DataManagementApplicationMasterEntitiesMaintain
- DataManagementApplicationMasterEntitiesView
- DataManagementApplicationParametersEntitiesMaintain
- DataManagementApplicationParametersEntitiesView
- DataManagementApplicationReferenceEntitiesMaintain
- DataManagementApplicationReferenceEntitiesView
- DataManagementApplicationTransactionEntitiesMaintain
- DataManagementApplicationTransactionEntitiesView

When you create data entities that can be used in the Data management workspace, you must extend these duties with the new security privileges, based on the Entity Category property that is specified on the data entity. (For information about how to extend duties with the new security privileges, see the "Privilege/duty mapping" section earlier in this topic.) You can also use the duties to create new roles for specific data management scenarios.

Modeling new entry point security in the Application Explorer

The pattern for modeling security resembles the pattern for modeling security with privileges on an entry point. To model security, follow these steps.

1. Create a new privilege.
2. Create new data entity permissions.

3. Set the Name to Data Entity.

4. Select the Access Level.

5. Select Integration Mode (All > Data Services > Data Management). This is specific to Object Type: Data Entity.

   - All – Applies same security settings to be applied to both OData and data import/export.
   - Data Management – Applies only to data import/export and connector integration.
   - Data Services – Only applies to OData Services.

Sensitive data

The Table Protection Framework (TPF) enables strict access control to data that is stored in Finance and Operations. This feature is exposed through the AOSAuthorization property on tables and table fields. If you mark a table or field by using AOSAuthorization, the security framework now requires that a user be granted explicit access to that resource. This requirement also applies when the table or field is accessed through data entities. This section describes the guidelines for granting TPF permissions for data entities.

Data management

For data entities that are targeted at data migration, you should assign TPF permissions to the corresponding import/export privileges. In this way, you help guarantee that all data can be imported and exported.

Integration by using OData

For data entities that are targeted at integration scenarios, the TPF permissions that you should assign depend on whether the TPF-protected field is essential for the data entity as a whole to work:

- **If the TPF-protected field is essential**: An essential field is a field that will always be read/written. In this case, TPF permissions should be granted to the same privileges that grant access to the data entity.

- **If the TPF-protected field isn’t essential**: Examples of nonessential fields include the field for a worker’s Social Security number and the field for a vendor’s bank account number. In this case, TPF permissions for accessing the field should be granted through a separate privilege, and that privilege should be assigned directly to the roles that require access to the TPF-protected field. However, if the field is a mapped field on the entity, that access has probably already been granted to the role, if that role also has access to the field through pages in the client UI.

There are several advantages to granting explicit access to TPF-protected fields that aren’t considered essential for the entity:

- You can more easily discover who has access to sensitive data.
- You help reduce the risk that someone will gain access to sensitive data by accident, because a role gains
access only if you assign both a duty and a privilege to it.
At [https://github.com/Microsoft/Dynamics-AX-Integration](https://github.com/Microsoft/Dynamics-AX-Integration), Microsoft provides sample code for consuming services. However, there are many scenarios where the other endpoint in an integration might not use a Microsoft stack. Even when the other endpoint does use, for example, the Open Data Protocol (OData) client code that Microsoft makes available, you might find it useful to perform the following actions:

- Explore and analyze how an interaction's messages are constructed.
- Test the response of a service to a well-known request.
- Determine how exceptions will appear to the other endpoint.

Many frequently used tools that will help you perform these actions are available. This topic isn't an endorsement of any tool. Although it provides examples that use some frequently used software utilities, the principles should broadly apply to other, similar tools.

### Prerequisites

Before you can test a service by using an external application, you must register the application in Microsoft Azure, and in Finance and Operations.

For details, see:

- [Register an application with AAD](#)
- [Register your external application](#)

### Query OData by using Postman

Postman ([https://www.getpostman.com/postman](https://www.getpostman.com/postman)) is a tool that is often used to interact with RESTful services (such as OData) in scenarios that involve the development and testing of application programming interfaces (APIs). This procedure isn't an endorsement of Postman, and other similar tools are available. However, we are using Postman to illustrate the concepts and messages that are involved when you use OAuth to authenticate with Azure AD, and then make OData requests to and receive responses from the application.

1. Start Postman.

2. In the upper-right corner, select the gear button, and then select **Manage environments** to create or update an environment.

3. Enter a name for the environment, and then select **Bulk Edit**.

4. Enter key-value pairs as shown in the following table. Enter one pair per line, and separate the key and value by using a colon (:).

<table>
<thead>
<tr>
<th>KEY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>tenant_id</td>
<td>The Azure tenant ID that you looked up during the setup of prerequisites</td>
</tr>
<tr>
<td>client_id</td>
<td>The Azure AD application ID that you registered during the setup of prerequisites</td>
</tr>
</tbody>
</table>
5. To verify that the key-value pairs can be parsed correctly, select **Key-Value Edit**, and review the results.

6. Close the environment page.

7. In the field to the left of the gear and eye buttons, select the new or updated environment.

8. To retrieve an Azure AD token, create a POST request that has a URL in the format `https://login.microsoftonline.com/[tenant ID]/oauth2/token`. You can use a URL parameter that refers to the `tenant_id` environment variable, such as `https://login.microsoftonline.com/:tenant_id/oauth2/token`.

9. On the **Body** tab, add body elements as request parameters that refer to the environment variables that you created earlier. Select **Bulk Edit**, enter the keys from the previous table, enter a colon (:), and then enter the key name again but enclose it in double braces ({{}}). Enter one request parameter per line. For example, enter `grant_type:{{grant_type}}`. Here is an example.

10. On the **Tests** tab, create a test that validates that the response is reasonable, and that stores the returned authorization token in an environment variable. Here is an example.
var json = JSON.parse(responseBody);
tests["Get Azure AD Token"] = !json.error && responseBody !== '' && responseBody !== '{}' && json.access_token !== '';
postman.setEnvironmentVariable("bearerToken", json.access_token);

11. Select **Save**, enter a name and collection for the request, and then select **Save** again.

12. Select **Send** to make the authorization request. The **Body** tab should now contain an Azure AD token together with other response details.

13. Because of the test code, the token is now in an environment variable. You can see that the token is an environment variable by selecting the **Environment quick look** button (the eye button).

14. Create a request to perform create, read, update, or delete (CRUD) operations on the desired data entity.
via the OData service. Create the URL according to your requirements. For more information, see Open Data Protocol (OData). You might find it useful to parameterize the request by using a variable that is stored in the environment, as shown earlier. The following example of a GET query uses a **Customer Account** parameter. The query returns name and address details for the customer account that is specified in the environment variable. Note that special characters must be correctly URL-encoded.

```javascript
https://[Finance and Operations instance URL]/data/Customers?
  $format=json&$filter=CustomerAccount%20eq%20'{custAccount}'&$select=CustomerAccount,Name,AddressDescription,FullPrimaryAddress
```

15. Add an Authorization header that refers to the authorization token that was retrieved earlier and stored in the **bearerToken** environment variable. The token must be prefixed by **Bearer** in the header.

![](image)

16. Create a test to help validate the response. The following example tests that non-empty, JSON-formatted data is returned in the response body.

```javascript
var json = JSON.parse(responseBody);
tests["Get customer info"] = !json.error && responseBody !== '{}' && responseBody !== '{}';
```

17. Save and send the request, and then verify the result. You must ensure that the user account being used is set to a default company that has data. Alternatively, you can also specify cross-company=true as the query parameter in the OData request.

![](image)
In our example, we have now successfully authenticated and then used the OData service to read a customer record.

Query the SOAP custom service in your application by using SoapUI

SoapUI (https://www.soapui.org/) is a tool that is often used to interact with SOAP and REST web services in scenarios that involve API development and testing. This procedure isn't an endorsement of SoapUI, and other similar tools are available. However, we are using SoapUI to illustrate the concepts and messages that are involved when you use OAuth to authenticate with Azure AD, and then make SOAP requests to and receive responses.

1. Start SoapUI, and select the SOAP button to create a project.

2. Complete the information for the project:
   - In the Project Name field, enter a name for the project.
   - In the Initial WSDL field, enter the service address, and add the suffix ?wsdl. (The service address should be in the format [Finance and Operations instance base URL]/soap/services/[service group name].) For more information, see the Services home page.
     For example, we are querying the user session service at the URL https://[Finance and Operations base URL]/soap/services/UserSessionService?wsdl.
   - Select the Create sample requests for all operations? check box.
     Because you selected to create sample requests, one sample request is created for each service operation that is available.

3. Right-click the new project, and then select New TestSuite to create a test suite. This test suite will generate a POST request for an Azure AD authorization token.

4. Right-click the test suite, and then select New TestCase.

5. Expand the test case, right-click Test Steps, select Add Step, and then select HTTP Request.

6. Enter a name for the request, and then select OK.

7. Enter a name for the test step. The endpoint that you should use for the POST request is
8. Use the plus sign (+) button next to Parameters to add the following values.

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>grant_type</td>
<td>client_credentials</td>
</tr>
<tr>
<td>client_id</td>
<td>The application ID from the Azure AD application registration</td>
</tr>
<tr>
<td>client_secret</td>
<td>The secret key value from the Azure AD application registration</td>
</tr>
<tr>
<td>resource</td>
<td>The URL of the instance without the trailing '/'</td>
</tr>
</tbody>
</table>

9. To make sure that the parameters are in the POST body, select Post QueryString, and then select Play. An access token should be returned in the response pane. The values will be most readable if you use the JSON response tab. Copy the access token so that you can use it in the authorization header of subsequent requests.

10. Go back to the first request node under the GetUserSessionInfo SOAP sample request. In the request pane on the left, select the plus sign (+) button to add a header that is named Authorization. Paste the access token into the Value field, and add the prefix Bearer.

11. The sample requests that SoapUI creates won't work unless you modify them. You must edit the call context and body so that they are consistent with the schema for what you're trying to do.

For our simple scenario, you can edit the optional call context elements so that they are null-valued. Insert a forward slash (/) before the greater than sign (>) in the opening tags. Then comment out the question marks (?) and the closing tags by using the standard <!--...--> syntax to delimit the start and end of the comments. (Question marks aren't valid content for the XML schema.) Alternatively, you can just delete the question marks (?) so that the context elements are empty.

12. The SOAP request is now ready. Select Play, and validate the result on the right.

In our example, we have now successfully authenticated and then queried UserSessionService via SOAP.
Create read-only entities that expose financial dimensions

In this topic, we describe how to build an entity for registered transactions that are registered.

NOTE

This topic comes from Per Baarsoe Jorgensen of the Solutions Architecture team. It describes a real-world scenario that we have encountered as we work with customers.

Imagine a scenario where we must expose all vendor invoice line transactions together with the financial dimensions that were applied through the distributions. Because easy consumption by a third-party tool is essential, we will create an entity for this scenario. As a result, the entity should not have to be joined with other related entities but should be able to provide value on its own.

We will walk through the process of creating a sample entity to meet these requirements. (We will leave out instructions for integrating with Microsoft Azure DevOps, because those steps are already well documented.)

Create a basic entity

The first step is to create a new element in a project by selecting New Item.

In the form that opens, under Data Model, we select the Data Entity element type.
NOTE

Be careful when you name the entity, because you can't rename it later.

Next, in the Data Entity wizard, we select the appropriate primary data source. For our scenario, this data source is VendInvoiceTrans.

The wizard doesn’t accept tables that don’t have a natural key, as is the case with VendInvoiceTrans. Therefore, we receive the following error message.
The workaround is to select any other primary data source that has a natural key associated, such as VendGroup.

Because we don’t really need this data source, we also don’t need any fields for it. Therefore, we clear the Select all check box.

Finally, we create the template for our entity by clicking Finish.

**Customize the basic entity**

The entity, staging table, and security assets have been created, and we can now produce our custom entity. In the project, we open the VendorInvoiceTransactionDetailsEntity entity in the designer.

In the designer, we replace the dummy table (VendGroup) that we applied in the wizard with the transaction table VendInvoiceTrans. Because we didn’t choose to add the fields, we don’t have to remove fields in the entity.
NOTE
Because we are exposing transactional data by using this entity, it’s important that we mark the entity as read-only. Therefore, we set the Is read only property to Yes on the top node. Because accounting distributions are versioned, it’s important that only the current version be returned when we query. Therefore, we create a view that makes this part easily reusable across multiple entities.

In the properties, we assign ReferenceDistribution a range filter value of 0 and ReferenceRole a range filter value of 1. The join mode property for the AccountDistributionReverse data source must be NoExistsJoin. After the new view is in place, we can add it to the VendorInvoiceTransactionDetailsEntity entity that we are currently building.

Expose financial dimensions as fields
The next important step is to expose the financial dimensions as separate fields on the entity. Because our
scenario builds on top of a posted transaction, we must add the fields to the DimensionCombinationentity entity. We want to make the adjustments in a resilient manner by using the extension approach, so that minimal maintenance will be required when we upgrade the code base to newer versions in the future.

**Microsoft Dynamics 365 for Finance and Operations, Enterprise edition version 1611**

For version 1611 or later, you should use the wizard that is available in Microsoft Visual Studio ([Dynamics 365 > Addins > Add financial dimensions for Odata](https://docs.microsoft.com/en-us/dynamics365/financials/add-finance-dimensions-odata)). For instructions, see [Add dimensions to Excel templates](https://docs.microsoft.com/en-us/dynamics365/financials/add-dimensions-excel).

**Earlier versions**

If you're working with earlier versions, you must complete the steps that are outlined here. First, we add the entity extension itself. Select **Create extension** on the context menu (shortcut menu). Next, we create the code that retrieves the data. Because the entity extension is already in place, we must create a new class. The following example adds code for an arbitrary dimension that is named **ProductLine**.

```java
[ExtensionOf(dataentityviewstr(DimensionCombinationentity))]
public final class DimensionCombinationentity_Extension
{
    private static server str getEmptyOrDimensionValueSqlString(str _attributeName)
    {
        str sqlStatement;
        DimensionAttribute dimensionAttribute = DimensionAttribute::findByName(_attributeName);
        if (!dimensionAttribute)
        {
            sqlStatement = SysComputedColumn::returnLiteral('');
        }
        else
        {
            sqlStatement = strFmt('SELECT TOP 1 T1.%1 ', dimensionAttribute.DimensionValueColumnName);
        }
        return sqlStatement;
    }
    /// Generates the sql to populate the FOTA view field.
    public static server str ProductLineValue()
    {
        return DimensionCombinationentity::getEmptyOrDimensionValueSqlString('ProductLine');
    }
}
```

We now add fields to the newly created entity extension by using custom fields that reference these methods.

![Image of entity extension](image)

Next, we set the property values to reflect the fact that the field is unmapped and should be retrieved through the code that we made for the extension class. When you create the relation, also set the following values:

- **Cardinality:** ZeroMore
- **Related data entity:** DimensionCombinationentity
- **Related data entity cardinality:** ZeroOne
- **Relationship type:** Association
**NOTE**

We must fully qualify the data method with the class name.

We are now ready to add the DimensionCombinationEntity entity to our new VendInvoiceTransactionEntity entity.

Notice that both the AccountingDistributionCurrent and the DimensionCombinationEntity entity should be outer-joined.
Now, we just have to drag the required fields from the various data sources to the **Fields** node on the new entity (that is, to our newly created ProductLine).

We should add a key to the entity to enable the incremental update functionality during the export configuration. Therefore, we must make sure that RecId from the VendInvoiceTrans data source is part of the fields named e.g. VendInvoiceTransRecId. After the field is in the field list, we can drag it to the **EntityKey** node.

To make sure that the staging table is associated with the fully configured entity, we must update it. On the context menu for the entity, we select **Update staging table**.

The entity work is now complete, and we can build it.

---

**NOTE**

In this scenario, a LedgerDimension was associated with the DimensionCombination entity entity. In scenarios where there is a DefaultDimension, we must associate it with the DimensionSet entity entity. The improvements and extensions that are required are identical to the improvements and extensions that we made to the DimensionCombination entity entity.
Additional resources

Export Dynamics AX 7 Entities to your own Azure SQL Database
Bring your own database (BYOD)

This topic explains how administrators can export data entities from the application into their own Microsoft Azure SQL database. This feature is also known as *bring your own database* (BYOD).

The BYOD feature lets administrators configure their own database, and then export one or more data entities that are available in the application into the database. (Currently, more than 1,700 data entities are available.) Specifically, this feature lets you complete these tasks:

- Define one or more SQL databases that you can export entity data into.
- Export either all the records (*full push*) or only the records that have changed or been deleted (*incremental push*).
- Use the rich scheduling capabilities of the batch framework to enable periodic exports.
- Access the entity database by using Transact-SQL (T-SQL), and even extend the database by adding more tables.

**Entity store or BYOD?**

If you followed the series of blog posts about Microsoft Power BI integration, you will be familiar with Entity store. Entity store is the operational data warehouse. Entity store provides built-in integration of operational reports with Power BI. Ready-made reports and analytical workspaces use Entity store. If you write Power BI reports by using data in your application environment, you should use Entity store.

However, the BYOD feature is recommended for the following scenarios:

- You must export data into your own data warehouse.
- You use analytical tools other than Power BI, and those tools require T-SQL access to data.
- You must perform batch integration with other systems.

**NOTE**

The application doesn't allow T-SQL connections to the production database. If you're upgrading from a previous version of Finance and Operations, and you have integration solutions that require direct T-SQL access to the database, BYOD is the recommended upgrade path.

You can use either Entity store or BYOD. The default operational reports that are available take advantage of embedded Power BI and Entity store. We recommend that you use our default operational reports as your first choice. You can also extend the ready-made operational reports to meet your requirements. You should consider BYOD a complementary option that you use as you require.

**Creating a SQL database**

Before you can configure the entity export option and use the BYOD feature, you must create a SQL database by using Azure portal.

For one-box development environments, you can create a database in the local Microsoft SQL Server database. However, this database should be used only for development and testing purposes. For production environments, you must create an Azure SQL database.

You should also create a SQL user account for sign-in to the database. Write down the server name, database
name, and the SQL user ID and password. You will use this information when you configure the entity export option in the next section.

If you’re using the BYOD feature for integration for analytical purposes, you should consider using clustered columnstore indexes as described in Columnstore indexes: Overview.

NOTE

Your BYOD database must be accessible to Finance and Operations apps. If you encounter issues where you are unable to access BYOD, you must ensure firewall rules in your BYOD are configured appropriately. For more information about self-service deployments, see Self-service deployment FAQ.

Selecting the correct service tier and compute size, is critical to secure expected performance. While doing this, it is important to consider the total, targeted workload and not just the load based on the Finance and Operations export. For production environments, it is recommended to use, at a minimum, compute size P2 in the Premium service tier or compute size S4 in the Standard service tier. Your specific BYOD usage might very well require a service tier greater than the above minimum. For more details about tiers and compute sizes, see SQL Azure service tiers and Detailed resource limits. To determine DTU needs or utilization, see Determine number of DTUs needed by a workload.

Configuring the entity export option

1. Start the client, and then, in the Data management workspace, select the Configure Entity export to database tile.

2. If you’ve configured any databases, a list is shown. Otherwise, you must configure a new database. In this case, select New, and then enter a unique name and a description for the new database. Note that you can export entities into multiple databases.

3. Enter the connection string in the following format:

   Data Source=<logical server name>,1433; Initial Catalog=<your DB name>; Integrated Security=False; User ID=<SQL user ID>; Password=<password>

   In this connection string, the logical server name should resemble nnnn.database.windows.net. You should be able to find the logical server name in Azure portal. The following illustration shows an example of a connection string.
The default extension field shown in the image above does not apply to BYOD.

4. Select **Validate**, and make sure that the connection is successful.

- The **Create clustered column store indexes** option optimizes the destination database for selected queries by defining columnstore indexes for entities that are copied.
- The **Enable triggers in target database** option sets export jobs to enable SQL triggers in the target database. This option lets you hook downstream processes into the trigger to orchestrate actions that must be started after records have been inserted. One trigger is supported per bulk insert operation. The size of the bulk insert is determined by the **Maximum insert commit size** parameter in the Data management framework.

For scenarios in which analytical applications data is read from BYOD, there is always the challenge of ensuring that the reporting systems get consistent data from BYOD while the sync is in progress. You can achieve this result by not reading the analytical data apps directly from the staging tables created by the BYOD process. The staging tables hold the data while data is being synced from the instance and hence will be constantly changing. Use the SQL trigger feature to determine when the data sync has been completed, and then transform and fill data to the downstream analytical data scenarios.

When the validation is passed, the database that you configured for entity export appears in lists of databases, as shown in the following illustration.

![Entity Store](image)

You can now publish one or more entities to the new database by selecting the **Publish** option on the menu.

**Publishing the entity schema to the database**

The **Publish** page enables several scenarios:

- Publish new entities to the database.
- Delete previously published entities from the database. (For example, you might want to re-create the schema.)
- Compare published entities with the entity schema. (For example, if new fields are added later, you can compare the fields with your database schema.)
- Configure change tracking functionality that enables incremental updates of your data.

The following sections discuss each option.

**Publish**

The **Publish** option defines the entity database schema on the destination database. When you select one or more entities, and then select the **Publish** option, a batch job is started. This job creates the entities in the destination database. When the database definition job is completed, you receive a message, which you can access by using the bell symbol in the upper right.

The actual data update occurs when you export data. At this point, you’re just creating the schema.

**Drop entity**
The **Drop entity** option deletes the data and the entity definition from the destination database.

**Compare source names**

The **Compare source names** option lets you compare the entity schema in the destination with the entity schema in the application. This option is used for version management. You can also use this option to remove any unwanted columns from the destination table.

**Configure change tracking**

Change tracking is a feature that is provided in SQL Server and SQL Database. Change tracking enables the database to track changes including deletes that are made on tables. The system uses change tracking to identify changes that are made to tables as transactions. However, because the application must track changes at the data entity level, there is additional logic on top of SQL change tracking to make this functionality work. The steps to enable change tracking are explained later in this section.

The **Change tracking** option on the Publish page lets you configure how changes are tracked on the underlying entity.

The following table describes the change tracking options that are available.

<table>
<thead>
<tr>
<th>OPTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable primary table</td>
<td>An entity consists of several tables. Select this option to track all changes that are made to the primary table of the entity. When changes are made to the primary table, the corresponding record is inserted into or updated in the destination database. Although data from the whole entity is written to the destination table, the system triggers the insert or update option only when the primary table is modified.</td>
</tr>
<tr>
<td>Enable entire entity</td>
<td>Select this option to track all changes to the entity. (These changes include changes to all the tables that make up the entity.) When changes are made to the entity, corresponding updates are made to the destination.</td>
</tr>
<tr>
<td>Enable custom query</td>
<td>This option lets a developer provide a custom query that the system runs to evaluate changes. This option is useful when you have a complex requirement to track changes from only a selected set of fields. You can also select this option when the entities that will be exported were built by using a hierarchy of nested views. For more information, see Enable change tracking for entities.</td>
</tr>
</tbody>
</table>
To use change tracking, you must enable the **Change tracking** option as shown above in data management. This action is available on the **Data entities** list page, by going to **Data management > Data entities**. You need to select an entity and select from one of the options listed above to enable change tracking on the data entity.

If you republish an entity that exists in the destination database, the system warns you that existing data will be deleted because of the new operation.

When you confirm the publish operation, the system publishes the schema to the database, and you're notified when the operation is completed.

By selecting the **Show published only** option on the **Publish** page, you can show only the entities that were published to a given destination database. The Publish function creates the entity schema in the database. You can navigate to the database and see the table schemas that were created, together with corresponding indexes.

---

**NOTE**

Currently, you can’t use BYOD to export composite entities into a database. You must export each entity in the composite entity.

---

### Exporting data into your database

After entities are published to the destination database, you can use the Export function in the **Data management** workspace to move data. The Export function lets you define a Data movement job that contains one or more entities.

You can use the **Export** page to export data into many target data formats, such as a comma-separated values (CSV) file. This page also supports SQL databases as another destination.

![Export page](image)

You can create a data project that has multiple entities. You can schedule this data project to run by using the batch framework. You also schedule the data export job to run on a periodic basis by selecting the **Export in batch** option.

### Exporting data across companies
Jobs that are executed in batch can also be used to export data across companies. This requires that the Enable all company export option is enabled under Data import/export framework parameters > Bring your own database. Concurrent exports for the same entity to the BOYD database may cause high DTU usage and can result in data loss for incremental exports. To avoid this risk, starting in version 10.0.16, all execution across companies will be sequential per company. This means that jobs with a high number of entities and companies take longer to run.

To reduce the overall export times, consider the following:

- Make sure that the same data entity is not in multiple projects so that the projects can run without conflicting with each other.
- Put entities that take a long time to run in separate projects. This will allow the other projects to run faster.
- Use Full push only exports instead of Incremental only exports for smaller data sizes. For example, you may want to do this if some of the entities have record counts of around 1,000 or less.
- Create cross-company entities if you do not need to export data per company, individually.

To create a cross-company entity:

1. Copy the current per company entity.
2. Change the name of PublicCollectionName and PubliceEntityName of the copied entity.
3. Add a column for DataAreaId, if needed.
4. Remove the value for the PrimaryCompanyContext property in order to not filter by the primary company during export.
5. Generate the staging table, and then build, and deploy the new entity.
6. Test the new entity to make sure that it works properly and performs adequately at required scale.
7. Schedule an export job in batch for the new entity, without selecting the Export across all companies option.

NOTE

Adding multiple entities to an export project for BYOD must be done carefully to ensure that the overall reliability of the BYOD export is not compromised. Different parameters must be taken into consideration when deciding the number of entities that are added to the same project. Some of these parameters should be the degree of complexity of the entities, data volume per entity that is expected, and the overall time for export to complete at the job level. Adding hundreds of entities must be avoided. Creating multiple jobs with a smaller number of entities is recommended.

Use of recurring exports in Manage > Manage recurring data jobs for BYOD is not supported. You must use the Export in batch option.

Incremental export

When you add an entity for data export, you can select to do an incremental export (which is also known as incremental push) or a full push. For incremental push to work, you must enable the Change tracking option in the database and specify an appropriate change tracking option, as described earlier in this topic.

NOTE

A full push deletes all existing records from an entity and then inserts the current set of records from the selected entity.

If you select an incremental push, the first push is always going to be a full push. This is because SQL needs to know which records have been ‘tracked’ in order to be able to track subsequent changes. Whenever a new record is inserted, or a record is added or deleted, the corresponding change will be reflected in the destination entity.
Because the first push is always a full push, we do not recommend that you do an explicit full push before you enable change tracking.

We recommend that you first enable change tracking and schedule an export job with incremental push. This will take care of the first full push and the subsequent incremental exports.

**Timeouts**

The default timeouts for BYOD exports are set to ten minutes for truncation operations and one hour for actual bulk insert operations. When volumes are high, these timeout settings may not be sufficient and must be updated. You can update the timeout settings by navigating to **Data management > Framework parameters > Bring your own database**. These timeouts are company specific and must be set separately for each company.

**Known limitations**

The BYOD feature has the following limitations.

**There should be no active locks on your database during synchronization**

Because BYOD is your own database, you must ensure that there are no active locks on your Azure SQL database when data is being synced. Having active locks on your database during synchronization can result in slow writes or even failure to export to your Azure SQL database.

**You can't export composite entities into your own database**

Currently, composite entities aren't supported. You must export individual entities that make up the composite entity which can be done in the same data project.

**Entities that don't have unique keys can't be exported by using incremental push**

You might face this limitation especially when you try to incrementally export records from a few ready-made entities. Because these entities were designed to enable the import of data, they don't have a unique key. However, you can enable change tracking only for entities that have a unique key. Therefore, there is a limitation on incremental push. One workaround is to extend the required entity and define a unique key.

**Troubleshooting**

**Incremental push not working correctly**

**Issue** - When a full push occurs for some entity then a large set of records can be seen in BYOD using a select statement. However, an incremental push results in only a few records in BYOD. It seems as if the incremental push deleted all the records and added only the changed records in BYOD.

**Solution** - In cases like this it is recommended to disable and re-enable change tracking for the entity in question. The state of the SQL change tracking tables might not be in the expected state. Also verify that there are no other incremental exports that cover the same tables (DMF, MR, Retail).

**SSIS Error Code DTS_E_OLEDBERROR. An OLE DB error has occurred. Error code: 0x80004005**

**Issue** - Export to BYOD fails with an SSIS exception shown below.

```
An OLE DB error has occurred. Error code: 0x80004005.
An OLE DB record is available. Source: "Microsoft SQL Server Native Client 11.0" Hresult: 0x80004005 Description: "Communication link failure".
An OLE DB record is available. Source: "Microsoft SQL Server Native Client 11.0" Hresult: 0x80004005 Description: "TCP Provider: An existing connection was forcibly closed by the remote host.
Failed to open a fastload rowset for <entityStaging>. Check that the object exists in the database.
OLE DB Destination failed the pre-execute phase and returned error code 0xC0202040.
```
**Solution** - This can occur if the connection policy on the Azure SQL BYOD server is set to Proxy. This must be changed to 'Redirect' as explained in [SQL DB Connectivity Architecture](#).
Automated Entity store refresh

11/24/2021 • 2 minutes to read • Edit Online

Overview

Entity store refresh is automated and managed by the system. Administrators do not need to schedule or monitor the Entity store refresh with the system batch schedules. The refresh operation is based on anticipated latency. This functionality is enabled in Platform update 23. As an administrator you do need to opt-in to use this feature.

Enable automated refresh

Complete the following steps to enable automated Entity store refresh.

1. Go to System administration > Set up > Entity store. On the Entity store page, a message indicates that you can switch to the Automated Entity store refresh option. This option is managed by the system. An admin does not have to schedule or monitor the Entity store refresh.

2. Select Switch now.

   **IMPORTANT**
   
   This action isn’t reversible. After you switch to the Automated Entity store refresh option, you can’t revert to the old user interface (UI) experience.

3. Select Yes to continue.

You will now see the new experience.

After the new experience is turned on, you can define the refresh for each aggregate measurement. The following refresh options are available:
- Every hour
- Twice a day
- Once a day
- Once a week

An admin can also refresh any aggregate measurement on demand by clicking the **Refresh** button. Additional options will be added in future platform updates. These options will include options for real-time refresh.

**IMPORTANT**

When automated refresh is enabled, the system can disable the refresh of aggregate measurements. You must revisit aggregate measurements and validate that appropriate refresh intervals have been applied.
Data task automation lets you easily repeat many types of data tasks and validate the outcome of each task. Data task automation is very useful for projects that are in the implementation phase. For example, you can automate the creation and configuration of data projects. You can also configure and trigger the execution of import/export operations, such as the setup of demo data and golden configuration data, and other tasks that are related to data migration. You can also create automated testing of data entities by using task outcome validation.

**IMPORTANT**

Data task automation isn’t currently supported for on-premises environments. The user who executes data task automation must be in the same tenant as the application environment and LCS project.

We recommend the following approach for data task automation.

1. **Identify the data-related tasks that will benefit from automation.**
   
   We recommend that implementation teams review their configuration management plan and data migration plan to identify potential data tasks for automation, and also to identify data entity test cases.

2. **Define tasks.**
   
   Tasks are defined in an XML manifest. You can keep your manifest under source control as part of configuration management in your application lifecycle management (ALM) strategy.

3. **Put the data packages that are related to data task automation in the Shared asset library of Microsoft Dynamics Lifecycle Services (LCS).** You can also use a specific LCS project as you require.

   Data task automation manager can consume packages from any sandbox and/or production environment that is related to the LCS project.

   **IMPORTANT**
   
   - The user account that runs Data task automation manager must have access to LCS and to the LCS project that is referenced in the manifest for data packages.
   - Although data task automation can be run in any environment in the cloud, we strongly recommend that you not run any import/export tasks that use integration application programming interfaces (APIs) in a production environment. Data task automation that involves integration APIs should be used only for automated testing.

4. **Run the data tasks, and then review the outcomes.**

   Data task automation manager provides the success or failure outcome for each task. It also provides insights into the reason why a task failed.
The following video is a 55-minute TechTalk that walks you through an early release of Task automation framework.

**Task manifest**

A task must be defined in an XML manifest. This section describes the manifest. For guidance about how to name and design the manifest, see the "Best practices for manifest design" section later in this topic.

**Manifest root**

The `<TestManifest>` element is the root of the manifest. All other elements are children of this element.

```xml
<xml version='1.0' encoding='utf-8'>
  <TestManifest name='Data management demo data set up'>
    <SharedSetup />
      <JobDefinition ID='ImportJobDefinition_1' />
        <EntitySetup ID='Generic' />
      </SharedSetup>
    <TestGroup />
  </TestManifest>
</xml>
```

<table>
<thead>
<tr>
<th>ELEMENT</th>
<th>ELEMENT CARDINALITY</th>
<th>ATTRIBUTES</th>
<th>ATTRIBUTE DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;TestManifest&gt;</code></td>
<td>1..1</td>
<td>name</td>
<td>The name helps to identify the purpose of the manifest.</td>
</tr>
</tbody>
</table>

**Shared setup**

The **Shared setup** section defines general task parameters and behaviors for all tasks in the manifest.

<table>
<thead>
<tr>
<th>PARENT ELEMENT</th>
<th>ELEMENT</th>
<th>ELEMENT CARDINALITY</th>
<th>ATTRIBUTES</th>
<th>ATTRIBUTE DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;TestManifest&gt;</code></td>
<td><code>&lt;SharedSetup&gt;</code></td>
<td>1..1</td>
<td>-</td>
<td>This element takes no attributes.</td>
</tr>
</tbody>
</table>

**Data files**

`<DataFile>` elements define the data packages and data files that the tasks in the manifest will use. The data files must be either in the LCS asset library of your LCS project or in the Shared asset library.

```xml
<DataFile ID='SharedSetup' name='Demo data-7.3-100-System and Shared' assetType='Data package' lcsProjectId='' />
<DataFile ID='FinancialsHQUS' name='Demo data-7.3-200-Financials-HQUS' assetType='Data package' lcsProjectId='' />
<DataFile ID='FinancialsPICH' name='Demo data-7.3-200-Financials-PICH' assetType='Data package' lcsProjectId='' />
<DataFile ID='FinancialsPIFB' name='Demo data-7.3-200-Financials-PIFB' assetType='Data package' lcsProjectId='' />
```

**IMPORTANT**

Although data task automation can be run in any environments in the cloud, we recommend that you not run any import/export tasks that use integration APIs in a production environment. Data task automation that involves integration APIs should be used only for automated testing.
### Data project definition

The `<JobDefinition>` element defines the data project definition. There can be more than one job definition in a manifest.

```xml
<JobDefinition ID='ImportJobDefinition_1'>
  <Operation>Import</Operation>
  <ConfigurationOnly>No</ConfigurationOnly>
  <Truncate></Truncate>
  <Mode>Import async</Mode>
  <BatchFrequencyInMinutes>1</BatchFrequencyInMinutes>
  <NumberOfTimesToRunBatch>2</NumberOfTimesToRunBatch>
  <UploadFrequencyInSeconds>1</UploadFrequencyInSeconds>
  <TotalNumberOfTimesToUploadFile>1</TotalNumberOfTimesToUploadFile>
  <SupportedDataSourceType>Package</SupportedDataSourceType>
  <ProcessMessagesInOrder>No</ProcessMessagesInOrder>
  <PreventUploadWhenZeroRecords>No</PreventUploadWhenZeroRecords>
  <UseCompanyFromMessage>Yes</UseCompanyFromMessage>
  <LegalEntity>DAT</LegalEntity>
  <PackageAPIExecute>true</PackageAPIExecute>
  <PackageAPIOverwrite>false</PackageAPIOverwrite>
  <PackageAPIReexecute>false</PackageAPIReexecute>
  <DefinitionGroupId>TestExport</DefinitionGroupId>
  <PackageName>TestExportPackage</PackageName>
</JobDefinition>
```
<table>
<thead>
<tr>
<th>PARENT ELEMENT</th>
<th>ELEMENT</th>
<th>ELEMENT CARDINALITY</th>
<th>ATTRIBUTE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;JobDefinition&gt;</td>
<td>&lt;Operation&gt;</td>
<td>1..1</td>
<td>-</td>
<td>The operation to be performed is specified by the following values: - Import - Export</td>
</tr>
<tr>
<td>&lt;Truncate&gt;</td>
<td>1..1</td>
<td>-</td>
<td>This is a Boolean field with possible values of Yes or No. This is applicable only when operation is set to Import.</td>
<td></td>
</tr>
<tr>
<td>&lt;Mode&gt;</td>
<td>1..1</td>
<td>-</td>
<td>The mode specifies the method using which the operation must be performed. The possible values are: - Import async - Export async - Recurring batch: This uses the enqueue API. Dequeue API is not supported yet. Package API supports both export and import.</td>
<td></td>
</tr>
<tr>
<td>&lt;ConfigurationOnly &gt;</td>
<td>0..1</td>
<td>-</td>
<td>This is a Boolean field with possible values of Yes or No. This must be set to Yes if the task is only to configure the data project but not to perform the specified operation.</td>
<td></td>
</tr>
<tr>
<td>&lt;BatchFrequencyInMinutes&gt;</td>
<td>1..1</td>
<td>-</td>
<td>This specifies the frequency in which the batch must be scheduled. This is applicable only when mode is set to recurring batch.</td>
<td></td>
</tr>
<tr>
<td>&lt;NumberOfTimesToRunBatch&gt;</td>
<td>1..1</td>
<td>-</td>
<td>This is used to set a limit to how many times the scheduled batch should run. This is applicable only when mode is set to recurring batch.</td>
<td></td>
</tr>
<tr>
<td>PARENT ELEMENT</td>
<td>ELEMENT</td>
<td>ELEMENT CARDINALITY</td>
<td>ATTRIBUTE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------------</td>
<td>---------</td>
<td>---------------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>&lt;UploadFrequencyInSeconds&gt;</td>
<td>1..1</td>
<td>-</td>
<td>This is used to control the rate at which a file is uploaded to the recurring batch job for import. This must be used only for automated testing of recurring integrations in non-production environments. This is applicable only when mode is set to <em>recurring batch</em> and operation is set to <em>Import</em>.</td>
</tr>
<tr>
<td></td>
<td>&lt;TotalNumberOfTimesToUpload&gt;</td>
<td>1..1</td>
<td></td>
<td>This controls the total number of times the file should be uploaded to the recurring batch. This must be used only for automated testing of recurring integrations in non-production environments. This is applicable only when mode is set to <em>recurring batch</em> and operation is set to <em>Import</em>.</td>
</tr>
<tr>
<td></td>
<td>&lt;SupportedDataSourceType&gt;</td>
<td>1..1</td>
<td></td>
<td>This must be used to specify if a file is being sent to the recurring batch or a package. This is only applicable when mode is set to ‘recurring batch’.</td>
</tr>
<tr>
<td></td>
<td>&lt;ProcessMessagesInOrder&gt;</td>
<td>1..1</td>
<td></td>
<td>This is a Boolean field with possible values of Yes or No. This is applicable only when mode is set to <em>recurring batch</em> and operation is <em>Import</em>.</td>
</tr>
<tr>
<td></td>
<td>&lt;PreventUploadWhenZeroRecords&gt;</td>
<td>1..1</td>
<td></td>
<td>This is a Boolean field with possible values of Yes or No. This is applicable only when mode is set to <em>recurring batch</em> and operation is <em>Export</em>.</td>
</tr>
<tr>
<td>PARENT ELEMENT</td>
<td>ELEMENT</td>
<td>ELEMENT CARDINALITY</td>
<td>ATTRIBUTE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------------</td>
<td>---------</td>
<td>---------------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>&lt;UseCompanyFromMessage&gt;</td>
<td>1..1</td>
<td></td>
<td>This is a Boolean field which can be set to Yes or No. This is applicable only when mode is set to recurring batch and operation is Import.</td>
</tr>
<tr>
<td></td>
<td>&lt;LegalEntity&gt;</td>
<td>1..1</td>
<td></td>
<td>This is used to specify the legal entity in which the import/export job must be executed.</td>
</tr>
<tr>
<td></td>
<td>&lt;PackageAPIExecute&gt;</td>
<td>1..1</td>
<td></td>
<td>Refer to package API documentation to understand this parameter. This is a Boolean field which takes 'true' or 'false'.</td>
</tr>
<tr>
<td></td>
<td>&lt;PackageAPIOverwrite&gt;</td>
<td>1..1</td>
<td></td>
<td>Refer to package API documentation to understand this parameter. This is a Boolean field which takes 'true' or 'false'.</td>
</tr>
<tr>
<td></td>
<td>&lt;PackageAPIReexecute&gt;</td>
<td>1..1</td>
<td></td>
<td>Refer to package API documentation to understand this parameter. This is a Boolean field which takes 'true' or 'false'.</td>
</tr>
<tr>
<td></td>
<td>&lt;DefinitionGroupID&gt;</td>
<td>1..1</td>
<td></td>
<td>Refer to package API documentation to understand this parameter. This is a string field.</td>
</tr>
<tr>
<td></td>
<td>&lt;PackageName&gt;</td>
<td>1..1</td>
<td></td>
<td>Refer to package API documentation to understand this parameter. This is a string field.</td>
</tr>
</tbody>
</table>

**Entity setup**

The **Entity setup** section defines the characteristics of an entity that a task in the manifest will use. There can be more than one definition, one for each entity that is used by the tasks in the manifest.
<EntitySetup ID='Generic'>
  <Entity name='*'>
    <SourceDataFormatName>Package</SourceDataFormatName>
    <ChangeTracking/>
    <PublishToBYOD/>
    <DefaultRefreshType>Full push only</DefaultRefreshType>
    <ExcelWorkSheetName/>
    <SelectFields>All fields</SelectFields>
    <SetBasedProcessing/>
    <FailBatchOnErrorForExecutionUnit>No</FailBatchOnErrorForExecutionUnit>
    <FailBatchOnErrorForLevel>No</FailBatchOnErrorForLevel>
    <DisableEntity>No</DisableEntity>
    <SkipStaging>Yes</SkipStaging>
    <ParallelProcessing>
      <Threshold/>
      <TaskCount/>
    </ParallelProcessing>
    <MappingDetail StagingFieldName='RoundingRulePrices' AutoGenerate='Yes' AutoDefault='No' DefaultValue=' ' IgnoreBlankValues='No' TextQualifier='No' UseEnumLabel='No'/>
  </Entity>
</EntitySetup>

<table>
<thead>
<tr>
<th>PARENT ELEMENT</th>
<th>ELEMENT</th>
<th>ELEMENT CARDINALITY</th>
<th>ATTRIBUTE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;SharedSetup&gt;</td>
<td>&lt;EntitySetup&gt;</td>
<td>1..n</td>
<td>ID</td>
<td>An identification that will be used by tasks to reference an entity definition to be used.</td>
</tr>
<tr>
<td>&lt;EntitySetup&gt;</td>
<td>&lt;Entity&gt;</td>
<td>1..1</td>
<td>name</td>
<td>The entity element is identified by the entity's name. However, to facilitate easy manifest definition, this element also supports * as a wild card which will mean all entities being used in a task. This comes in very handy when using data packages with hundreds of entities in a task.</td>
</tr>
<tr>
<td>&lt;Entity&gt;</td>
<td>&lt;SourceDataFormat Name&gt;</td>
<td>1..1</td>
<td>-</td>
<td>This is the file format to be used for the entity.</td>
</tr>
<tr>
<td></td>
<td>&lt;ChangeTracking&gt;</td>
<td>1..1</td>
<td>-</td>
<td>This is a Boolean field with possible values of Yes or No. It enables or disables change tracking on the entire entity.</td>
</tr>
<tr>
<td></td>
<td>&lt;PublishToBYOD&gt;</td>
<td>1..1</td>
<td>-</td>
<td>This is a Boolean field with possible values of Yes or No.</td>
</tr>
<tr>
<td>PARENT ELEMENT</td>
<td>ELEMENT</td>
<td>ELEMENT CARDINALITY</td>
<td>ATTRIBUTE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------------</td>
<td>---------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>&lt;DefaultRefreshType&gt;</code></td>
<td></td>
<td>1..1</td>
<td>-</td>
<td>This sets the default refresh rate on the entity. The possible values are Incremental push only or Full push.</td>
</tr>
<tr>
<td><code>&lt;ExcelWorkSheetName&gt;</code></td>
<td></td>
<td>1..1</td>
<td>-</td>
<td>This is used to specify the worksheet to be used for the entity.</td>
</tr>
<tr>
<td><code>&lt;SelectFields&gt;</code></td>
<td></td>
<td>1..1</td>
<td>-</td>
<td>This can be used to specify the fields to be included in the template for an export operation.</td>
</tr>
<tr>
<td><code>&lt;SetBasedProcessing&gt;</code></td>
<td></td>
<td>1..1</td>
<td>-</td>
<td>This is a Boolean field with possible values of Yes or No. It is used to enable or disable set based processing on an entity.</td>
</tr>
<tr>
<td><code>&lt;FailBatchOnErrorForExecutionUnit&gt;</code></td>
<td></td>
<td>1..1</td>
<td>-</td>
<td>This is a Boolean field with possible values of Yes or No. It is used to enable or disable failure at execution unit level on an entity.</td>
</tr>
<tr>
<td><code>&lt;FailBatchOnErrorForLevel&gt;</code></td>
<td></td>
<td>1..1</td>
<td>-</td>
<td>This is a Boolean field with possible values of Yes or No. It is used to enable or disable failure at execution level on an entity.</td>
</tr>
<tr>
<td><code>&lt;DisableEntity&gt;</code></td>
<td></td>
<td>1..1</td>
<td>-</td>
<td>This is a Boolean field with possible values of Yes or No. It is used to enable or disable an entity in a data project.</td>
</tr>
<tr>
<td><code>&lt;SkipStaging&gt;</code></td>
<td></td>
<td>1..1</td>
<td>-</td>
<td>This is a Boolean field with possible values of Yes or No. It is used to skip staging table for an entity during exports.</td>
</tr>
<tr>
<td>PARENT ELEMENT</td>
<td>ELEMENT</td>
<td>ELEMENT CARDINALITY</td>
<td>ATTRIBUTE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------------</td>
<td>---------</td>
<td>---------------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>&lt;ParallelProcessing&gt;</td>
<td>1..1</td>
<td>-</td>
<td>This is used to define the parallel processing set up for an entity. The task will delete these settings if already exits at the beginning of the task and it will delete the created settings at the end of its execution.</td>
</tr>
<tr>
<td>&lt;ParallelProcessing&gt;</td>
<td>&lt;Threshold&gt;</td>
<td>1..1</td>
<td>-</td>
<td>This specifies the threshold for the parallel processing rule.</td>
</tr>
<tr>
<td>&lt;TaskCount&gt;</td>
<td>1..1</td>
<td>-</td>
<td>This is used to specify the number of parallel tasks to be used for parallel processing.</td>
<td></td>
</tr>
<tr>
<td>&lt;Entity&gt;</td>
<td>&lt;MappingDetail&gt;</td>
<td>0..n</td>
<td>-</td>
<td>Allows to configure the auto generate, auto default and other settings on the mapping for an entity.</td>
</tr>
<tr>
<td>&lt;MappingDetail&gt;</td>
<td>-</td>
<td>StagingfieldName</td>
<td>This attribute is used to identify the entity column for which the settings are to be specified.</td>
<td></td>
</tr>
<tr>
<td>&lt;MappingDetail&gt;</td>
<td>-</td>
<td>AutoGenerate</td>
<td>This is a Boolean field with possible values of Yes or No for enabling/disabling auto generate option.</td>
<td></td>
</tr>
<tr>
<td>&lt;MappingDetail&gt;</td>
<td>-</td>
<td>AutoDefault</td>
<td>This is a Boolean field with possible values of Yes or No for enabling/disabling auto default option.</td>
<td></td>
</tr>
<tr>
<td>&lt;MappingDetail&gt;</td>
<td>-</td>
<td>DefaultValue</td>
<td>This is the default value to be used if auto defaulting is enabled.</td>
<td></td>
</tr>
</tbody>
</table>
Test groups

Test groups can be used to organize related tasks in a manifest. There can be more than one test group in a manifest.

```xml
<TestGroup name='Set up Financials'>
  <TestCase Title='Import shared set up data package' ID='3933885' RepeatCount='1' TraceParser='off' TimeOut='20'>
    <DataFile RefID='SharedSetup' />
    <JobDefinition RefID='ImportJobDefinition_1' />
    <EntitySetup RefID='Generic' />  
  </TestCase>
  <TestCase Title='Import financials for HQUS' ID='3933886' RepeatCount='1' TraceParser='off' TimeOut='20'>
    <DataFile RefID='FinancialsHQUS' />
    <JobDefinition RefID='ImportJobDefinition_1'>
      <LegalEntity>HQUS</LegalEntity>
    </JobDefinition>
    <EntitySetup RefID='Generic' />
  </TestCase>
  <TestCase Title='Import financials for PICH' ID='3933887' RepeatCount='1' TraceParser='off' TimeOut='20'>
    <DataFile RefID='FinancialsPICH' />
    <JobDefinition RefID='ImportJobDefinition_1'>
      <LegalEntity>PICH</LegalEntity>
    </JobDefinition>
    <EntitySetup RefID='Generic' />
  </TestCase>
  <TestCase Title='Import financials for PIFB' ID='3933888' RepeatCount='1' TraceParser='off' TimeOut='20'>
    <DataFile RefID='FinancialsPIFB' />
    <JobDefinition RefID='ImportJobDefinition_1'>
      <LegalEntity>PIFB</LegalEntity>
    </JobDefinition>
    <EntitySetup RefID='Generic' />
  </TestCase>
</TestGroup>
```
<table>
<thead>
<tr>
<th>PARENT ELEMENT</th>
<th>ELEMENT</th>
<th>CARDINALITY</th>
<th>ATTRIBUTES</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;TestManifest&gt;</td>
<td>&lt;TestGroup&gt;</td>
<td>1..n</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>&lt;TestGroup&gt;</td>
<td>Name</td>
<td>1..1</td>
<td></td>
<td>This is the name for the group to identify its functional reason.</td>
</tr>
<tr>
<td>&lt;TestGroup&gt;</td>
<td>&lt;TestCase&gt;</td>
<td>1..n</td>
<td>-</td>
<td>The task is defined in this element. The task can refer to the shared set up to inherit task parameters and task behavior. The task can also override parameters and behavior at its level thus making the management of the manifest simple.</td>
</tr>
<tr>
<td>&lt;TestCase&gt;</td>
<td>Title</td>
<td>-</td>
<td></td>
<td>This is the title for the task.</td>
</tr>
<tr>
<td>&lt;TestCase&gt;</td>
<td>ID</td>
<td>-</td>
<td></td>
<td>This is the ID for the task. This can be alphanumeric with a max character limit of 10.</td>
</tr>
<tr>
<td>&lt;TestCase&gt;</td>
<td>RepeatCount</td>
<td>-</td>
<td></td>
<td>This is a placeholder for a future functionality. However, this must be specified with a value of 1.</td>
</tr>
<tr>
<td>&lt;TestCase&gt;</td>
<td>TraceParser</td>
<td>-</td>
<td></td>
<td>This is a placeholder for a future functionality. However, this must be specified with a value off.</td>
</tr>
<tr>
<td>&lt;TestCase&gt;</td>
<td>Timeout</td>
<td>-</td>
<td></td>
<td>This is the maximum duration a task will be monitored by the task automation manager. If the task is still active beyond the timeout specified, the manager will proceed to the next task in the manifest.</td>
</tr>
</tbody>
</table>
### Best practices for manifest design

You can define a manifest in many ways. Here are a few pointers that you should consider when you design a manifest.

**Granularity**

We recommend that you determine the granularity of your manifest as a functional decision. Your team must...
Inheritance

The manifest schema supports inheritance of common elements that will apply to all tasks in the manifest. A task can override a common element to define a unique behavior. The purpose of the Shared setup section is to minimize repetition of configuration elements, so that elements are reused as much as possible. The goal is to keep the manifest concise and clean, to improve maintenance and readability.

Source control

Manifests that must be used by all the members of an implementation team should be stored in source control in the Application Object Tree (AOT). This approach not only provides the benefits of source control, but also enables a process to distribute or make manifests available to all users in a consistent manner. This approach also enables configuration management for data projects that are related to data management, if manifests are used for configuration.

Number of job definitions and entity definitions

For most of the use cases, one job definition in a manifest should be enough, because inheritance can be used to change the behavior at the task level. This principle also applies to entity definitions.

Validations

Data task automation manager performs validations, based on the setup of a task. If a task fails, you can quickly determine the reason for the failure by viewing the validations after the task is completed. The level of information that Data task automation manager provides is optimized to facilitate initial discovery. For detailed investigation, you must look at the corresponding data project and its execution details.

The following data validations are currently supported:

- **Job status** – This validation checks whether the job was successful.
- **Batch status** – This validation checks whether the batch was successful.
- **Message status** – If the test is about integrations, the message status is validated.
- **Truncation** – If truncation is enabled, this validation checks whether truncation occurred.
- **Skip staging** – If Skip staging is enabled on a test, this validation checks whether staging was skipped.

Example 1: Configuration management for data projects

The <ConfigurationOnly> element can be used to create configuration tasks for data projects. When ConfigurationOnly is set to Yes, the data projects are only created but not executed. This allows for managing data projects across environments in an automated manner.

```xml
<?xml version='1.0' encoding='utf-8'?>
<TestManifest name='Data management demo data set up'>
  <SharedSetup>
    <DataFile ID='SharedSetup' name='Demo data-7.3-100-System and Shared' assetType='Data package' lcsProjectId=''/>
    <DataFile ID='FinancialsHQUS' name='Demo data-7.3-200-Financials-HQUS' assetType='Data package' lcsProjectId=''/>
  </SharedSetup>
</TestManifest>
```
<JobDefinition ID='ImportJobDefinition_1'>
  <ConfigurationOnly>Yes</ConfigurationOnly>
  <Operation>Import</Operation>
  <Truncate>No</Truncate>
  <Mode>Import async</Mode>
  <BatchFrequencyInMinutes>1</BatchFrequencyInMinutes>
  <NumberOfTimesToRunBatch>2</NumberOfTimesToRunBatch>
  <UploadFrequencyInSeconds>1</UploadFrequencyInSeconds>
  <TotalNumberOfTimesToUploadFile>1</TotalNumberOfTimesToUploadFile>
  <SupportedDataSourceType>Package</SupportedDataSourceType>
  <ProcessMessagesInOrder>No</ProcessMessagesInOrder>
  <PreventUploadWhenZeroRecords>No</PreventUploadWhenZeroRecords>
  <UseCompanyFromMessage>Yes</UseCompanyFromMessage>
  <LegalEntity>DAT</LegalEntity>
</JobDefinition>

<TestCase Title='Set up import job for shared set up data package' ID='3933885' RepeatCount='1'
TraceParser='off' TimeOut='20'>
  <DataFile RefID='SharedSetup' />
  <JobDefinition RefID='ImportJobDefinition_1' />
  <EntitySetup RefID='Generic' />
</TestCase>

<TestCase Title='Set up import job for financials HQUS' ID='3933886' RepeatCount='1'
TraceParser='off' TimeOut='20'>
  <DataFile ID='FinancialsHQUS' />
  <JobDefinition ID='ImportJobDefinition_1'>
    <LegalEntity>HQUS</LegalEntity>
  </JobDefinition>
  <EntitySetup ID='Generic' />
</TestCase>

<TestCase Title='Set up import job for financials PICH' ID='3933887' RepeatCount='1'
TraceParser='off' TimeOut='20'>
  <DataFile ID='FinancialsPICH' />
  <JobDefinition ID='ImportJobDefinition_1'>
    <LegalEntity>PICH</LegalEntity>
  </JobDefinition>
  <EntitySetup ID='Generic' />
</TestCase>

<TestCase Title='Set up import job for financials PIFB' ID='3933888' RepeatCount='1'
TraceParser='off' TimeOut='20'>
  <DataFile ID='FinancialsPIFB' />
  <JobDefinition ID='ImportJobDefinition_1'>
    <LegalEntity>PIFB</LegalEntity>
  </JobDefinition>
  <EntitySetup ID='Generic' />
</TestCase>

<TestCase Title='Set up import job for financials HQUS' ID='3933886' RepeatCount='1'
TraceParser='off' TimeOut='20'>
  <DataFile ID='FinancialsHQUS' />
  <JobDefinition ID='ImportJobDefinition_1'>
    <LegalEntity>HQUS</LegalEntity>
  </JobDefinition>
  <EntitySetup ID='Generic' />
</TestCase>
Example 2: Automated setup of demo data

The following manifest shows the setup of demo data for three legal entities when the demo data packages are stored in the Shared asset library. The difference in this example from the previous example is the actual execution of the data projects to set up the demo data. This is accomplished by not using the ConfigurationOnly option or setting it to No to use it for consistency of the manifest.
<TestManifest>

  <TestGroup>
    <TestCase Title='Import shared set up data package' ID='3933885' RepeatCount='1' TraceParser='off'
      TimeOut='20'>
      <DataFile RefID='SharedSetup' />
      <JobDefinition RefID='ImportJobDefinition_1' />
      <EntitySetup RefID='Generic' />
    </TestCase>

    <TestCase Title='Import financials for HQUS' ID='3933886' RepeatCount='1' TraceParser='off'
      TimeOut='20'>
      <DataFile RefID='FinancialsHQUS' />
      <JobDefinition RefID='ImportJobDefinition_1'>
        <LegalEntity>HQUS</LegalEntity>
      </JobDefinition>
      <EntitySetup RefID='Generic' />
    </TestCase>

    <TestCase Title='Import financials for PICH' ID='3933887' RepeatCount='1' TraceParser='off'
      TimeOut='20'>
      <DataFile RefID='FinancialsPICH' />
      <JobDefinition RefID='ImportJobDefinition_1'>
        <LegalEntity>PICH</LegalEntity>
      </JobDefinition>
      <EntitySetup RefID='Generic' />
    </TestCase>

    <TestCase Title='Import financials for PIFB' ID='3933888' RepeatCount='1' TraceParser='off'
      TimeOut='20'>
      <DataFile RefID='FinancialsPIFB' />
      <JobDefinition RefID='ImportJobDefinition_1'>
        <LegalEntity>PIFB</LegalEntity>
      </JobDefinition>
      <EntitySetup RefID='Generic' />
    </TestCase>
  </TestGroup>
</TestManifest>
This topic provides an overview of the Data validation checklist workspace and the associated configuration.

The Data validation checklist workspace lets you track data validation processes across companies, areas, and people. The checklist can be used during a new implementation, after an upgrade, or after a migration. Depending on your view of the Data validation checklist workspace, you'll see either all tasks and statuses for a data validation project, or just the tasks that are assigned to you.

You must first select a data validation project at the top of the workspace. All data that is shown in the workspace is then filtered by the selected data validation project.

Summary tiles

The Summary tiles provide an overview of the process, and indicators help you keep the data validation process on track. You can see all remaining tasks, completed tasks, in progress tasks, and not started tasks for the process. This information is for all companies that are included in the selected data validation project.

Tasks and status section

In the Tasks and status section, the status of the overall data validation project is displayed in various ways: status by legal entity, by area, and by task list. You can select the filter to view the status for a specific company. Each status tab provides a breakdown by both the percentage that has been completed and the number of tasks that remain.

The last tab is for the detailed task list. This list shows the full task list. You can filter the task list in several ways. Click Edit task to change the status of a task or assign a task. Click Attachments to view attachments for a task.

The task name is a hyperlink to the page where the user must go to complete the work. You can set this hyperlink by using the Menu item name field when you edit or create a task from the Configure data validation project form.

You can attach files, notes, images, and URLs to a task by using the Attachments action. For example, you can attach a report file that was printed for a task. An icon appears in the Attachment column for the task if an attachment is present.

The Completed by option will be automatically filled after the task is completed with the name of the worker who completed the task. When a task is marked as completed, the Completed date field is automatically updated to the current date and time.

Configure data validation project page

Before you can use the Data validation checklist workspace, you must configure the process by using the Configure data validation project page. (Click Workspaces > Data validation checklist > Configure data validation project.)

Task areas

You use task areas to group data validation tasks into logical areas of ownership within your organization. For
example, Accounts payable, Accounts receivable, or General ledger might be used as task areas.

The **Menu item name** is associated with the task work effort and can be used to go directly to the associated page from the task link in the workspace. For example, a data validation task to run the **Accounts payable aging** report for Accounts payable can be linked to the **Accounts payable aging report** page.
This topic documents the scenarios when a specific error will be seen. These scenarios aren’t a complete list of errors and scenarios, however this list will be continuously updated so keep checking back for updates. Any feedback on this page with regards to specific errors that should be covered are welcome.

This topic describes the error messages that you might encounter in data management.

**Import to target failed due to an update conflict as more than one process is trying to update the same record at the same time**

When you use recurring imports (enqueue API), if the files are sent to the end point at high frequency and the sequential processing of messages isn’t enabled, data management will try to process the files in parallel. When files are processed in parallel, and multiple files have the same record, multiple threads will try to update the same record at the same time. If this is a data issue, you must update the data so that the same records don’t repeat across files. If this is not a data issue and the entity is expected to handle such cases, this might be a bug. For bugs, to mitigate the issue, you can choose to sequentially process the files. If this is not a data issue and the entity is not expected to process in parallel, then this entity must not be subjected to parallel processing. You should enable sequential processing of messages in the recurring job.

**There are field(s) which are not mapped to Entity <EntityName>**

It is a common practice to use the export functionality to generate the entity template file that can be later used for imports. However, while exporting the template, in fixed width format with ‘First row header’ set to ‘No’ (in source data formats set up), the exported template will not have the column names. When this file is imported, it will result in this error.

**Data package download - Error downloading data package for job "". Record for ID - {GUID} not found**

One of the scenarios where this error can happen is when the environment, such as the dev environment, points to the database in another environment, such as UAT, and the export job is run from the source environment which is dev in this example. The exported file gets uploaded to the blob storage that is associated with the source environment (dev, in this example). However, this job will also show up in the target environment (UAT) since the database is shared. If you try to download the exported file using the Download file option, this error will display because the file does not exist in the blob storage of the target environment (UAT) from where you are trying to download.

**XML is not in correct format-Exception from HRESULT: 0xC0010009**

This message covers all XML formatting issues in the file. For example, the data project has mappings for columns that do not exist in the file that is being used for the operation. This error can happen if certain columns were removed from the file and this file is now used. Either fix the mapping in the data project or fix the file to have all the columns as expected.

**Error while uploading a file during export**

When running an export on a development environment, an error could occur relating to not being able to
upload the export file. This could occur if Azure Storage Emulator is not available or an older version of the emulator is installed. To resolve this issue, install the latest emulator, restart the virtual machine (VM), and rerun the export job. The storage emulator can be installed from Azure Storage Emulator.
What is Azure Data Lake?

Microsoft Azure Data Lake is a technology in Azure cloud that enables big data analytics and artificial intelligence (AI). When this topic mentions "Data Lake," it's referring specifically to storage technology that is based on Azure Data Lake Storage Gen2.

Data lakes provide cloud storage that is less expensive than the cloud storage that relational databases provide. Therefore, large amounts of data can be stored in the cloud. This data includes both business data that is traditionally stored in business systems and data warehouses, device and sensor data, such as signals from devices. In addition, Data Lake supports a range of tools and programming languages that enable large amounts of data to be reported on, queried, and transformed.

For an overview of Data Lake Storage Gen2, see Introduction to Azure Data Lake Storage Gen2.

Dynamics 365 products, such as Finance and Operations apps, use Data Lake for AI and analytics scenarios. Therefore, customers can take advantage of the strengths and cost advances that this technology offers. The following sections provide an overview of the scenarios.

**Analytical workspaces**

Analytical workspaces provide contextual and actionable insights in Finance and Operations apps. They give users a bird’s-eye view of a business process, so that they can immediately get relevant information and take appropriate action.

Analytical workspaces are based on Entity store and use embedded Power BI technology to provide rich, interactive visuals of data from Finance and Operations apps. Analytical workspaces are fun and exciting to use: they invite your users to explore data.

Analytical workspaces can be used for operational analytics scenarios in two ways:

- Use and extend the ready-made analytical workspaces, so that you don’t have to build workspaces from scratch.
- Build your own Power BI–based analytical reports.

For more information, see Embedded Power BI in workspaces.

**BYOD**

Bring your own database (BYOD) is a service that lets customers extract data from Finance and Operations apps into their own data warehouses. We recommend that you use BYOD when you must combine data from Finance and Operations apps with other systems or with reporting that uses earlier data.

For more information, see Bring your own database (BYOD).

**Data Lake combines BYOD and Entity store**

Customers use a combination of analytical workspaces (which are based on Entity store) and BYOD for different scenarios. Following table compares the scenarios and capabilities
Near real-time Operational Analytics (with embedded Power BI)

Reporting on data from F&O

Ready-made reports shipped with F&O or ISV solution

Data warehouse with PowerBI.com or other tools

Mash-up F&O data with other data sources

Reports extended or authored by partner

Reports authored with both these sources can be pinned into Analytical workspaces in F&O with contextual security and drill thru actions. Data Lake combines both these services into a single service that offers the “best of both worlds”:

- Because Data Lake is included in customer subscriptions, you can bring your own data lake and integrate it with Finance and Operations apps. Finance and Operations apps will use your data lake to store Entity store data and operate analytical workspaces. Analytical workspaces continue to work as they worked before.

- Entity store is staged in your data lake and provides a set of simplified (denormalized) data structures to make reporting easier. Your users can now be given direct access to the data that is most relevant to them, and they can create their own reports by using a tool of their choice.

- Instead of exporting data by using BYOD, customers can select the data that is staged in the data lake. Data feed service, which is part of Finance and Operations services, keeps the data in the data lake fresh.

- You can bring your own data into the data lake to supplement the data that Finance and Operations apps provide. This capability allows for easy data mash-up scenarios in the data lake.
  - Data from external sources can easily be ingested into the data lake via hundreds of ready-made connectors that are available in tools such as Power BI dataflows and Azure Data Factory.
  - Historical data and earlier data that is often inherited as a part of the transition to Finance and Operations apps can be ingested directly into the data lake.
  - Data lakes provide options for ingesting non-business data. For example, device data can easily be ingested into the data lake.

- Cloud-based services let both power users and developers consume this data.

Cloud-based services let both power users and developers consume this data.

Common Data Model folders

Data is stored in Data Lake to comply with the Common Data Model folder standard. Here are some results:
Data that Finance and Operations apps stage in Data Lake is organized into a set of folders. Common Data Model folders contain metadata definitions in addition to data files. Metadata definitions are kept in model files, according to the standard that is specified by the Common Data Model language. Because metadata is present and data storage complies with the Common Data Model folder standard, Azure and other services can read and transform the data.

The following illustration shows the Common Data Model folder structure from Finance and Operations apps.

For more information about Common Data Model in Data Lake, see Use the Common Data Model to optimize Azure Data Lake Storage Gen2.

Here is an example:

- You can attach a Common Data Model folder to Power BI dataflows as a reference dataflow. You can work with Power BI dataflows and further reshape the data, or you can create Power BI datasets and reports.
- You can use Data Factory or other data transformation tools to further shape the data.

Like Finance and Operations apps, other services (including Dataverse), Azure IoT, and many third-party tools and service can understand and work with data in Common Data Model folders. The list of services is growing. Here are some examples:

- Dataverse lets you export data to your own data lake. For more information, see Exporting Dataverse data to Azure Data Lake is Generally Available.
- Power users can transform data in Data Lake by using Dataverse dataflows. For more information, see Use the Common Data Model to optimize Azure Data Lake Storage Gen2.

How you can use Data Lake later if you're currently using BYOD

In most cases, customers use BYOD to extract data from Finance and Operations apps so that they can use that data for reporting or analytics. BYOD requires that customers provision and maintain an Azure SQL database to store data that is exported from Finance and Operations apps.

Some customers use the exported data in BYOD for reporting. These customers can just point reporting tools to
Because data is already present, export isn't required

Data Lake integration lets users select tables and entities, just as they can in the BYOD experience. After tables and entities are selected, the system updates the data in Data Lake. The system also continuously exports data as it changes in Finance and Operations apps. The data lake reflects the updated Finance and Operations data within a few minutes after the changes occur.

**NOTE**

Table data is updated within minutes after a change occurs in Finance and Operations apps. According to the current service-level agreement (SLA) that the services offer, data is updated within 10 minutes.

Because of Data Lake integration, customers don't have to monitor and manage complex data export and orchestration schedules. No user intervention is required to update data in the data lake.

Reduced cost of data storage

Data is stored in a data lake (Gen2) instead of the SQL database that BYOD requires. Therefore, customer can use a storage medium that is much less expensive than Azure SQL Database.

**NOTE**

Because Data Lake Storage Gen2 is included in a customer's subscription, the customer must pay for data storage and input/output (I/O) costs that are incurred when data is read and written to the data lake. The customer might also incur I/O costs because Finance and Operations apps write data to the data lake or update the data in it. To help reduce intra-region I/O costs, Finance and Operations apps require that data lakes be provisioned in the same country or region as the Finance and Operations environment.

For more information about cost, see the [Azure Data Lake Storage Gen2 pricing](#) page.

Existing downstream/consumption pipelines can be preserved

As was discussed earlier, BYOD is mostly used in two scenarios:

- Reporting tools and other tools access BYOD directly.
- BYOD is used as a temporary staging area to store data while it's being exported to other downstream systems, such as data warehouses.

For the first scenario, you can point reporting tools to the SQL database. Many reporting tools work with SQL databases, because they can use Transact-SQL (T-SQL) to read data.

For the second scenario, you can use data integration/transformation tools such as Data Factory. Many data integration tools can consume data directly from the data lake.

For both scenarios, if you're using T-SQL to read the database, you can create a SQL Server endpoint by using Azure Synapse Analytics. Azure Synapse include SQL-on-demand capability that enables Data Lake to be
queried by using the T-SQL language. Downstream tools don't have to be modified, because you can preserve the data shape, as you can when you use BYOD.

**Simplified data pipeline for near-real-time reporting**

In a traditional data warehouse, data is stored in a staging area before it can be aggregated and simplified for reporting. You might also have reporting tools that aggregate data for better user experiences. If you have multiple data stops (for staging, denormalization, and aggregation), data staleness increases (that is, after activity occurs, more time passes before reports reflect the results).

Traditional data warehouses that have multiple data stops are ideal for reporting on data that changes every day or even several times a day. However, if you must report on near-real-time data (for example, if you must report on retail sales data within minutes after the sales occur), you might have to design a data pipeline that has minimal data stops. This pattern is known as *hot-path, cold-path reporting* (or *lambda architecture*). Multiple data stops and multiple data pipelines increase complexity and management efforts.

Data Lake integration makes *warm-path reporting* available as the default reporting option. Because data in a data lake is updated within minutes, this approach might be acceptable for most reporting scenarios, even scenarios that involve near-real-time reporting.

For near-real time reporting, you can minimize the data staging and preparation steps by using on-demand query (for example, Azure Synapse) and Power BI direct query mode, which queries semi-prepared data in the data lake. For analytical reporting, you can denormalize and aggregate the data in the data lake.

**How you can use Data Lake later if you're currently using analytical workspaces**

Analytical workspaces make in-context operational analytics available in Finance and Operations apps. Analytical workspaces are built as an extension to Finance and Operations workspaces. They are a type of a "cockpit" that provides a bird's-eye view of business processes. There are more than 25 ready-made analytical workspaces that provide interactive, near-real-time data exploration.

You can drive action by adding contextual gestures to analytical workspaces. Therefore, users can act on results without having to leave the report that they are viewing. An extensive programming model that is built into Finance and Operations apps enables enterprise resource planning (ERP) actions and business logic to be driven from analytical workspaces.

You can use the ready-made analytical workspaces from Microsoft, partners, and independent software vendors (ISVs) as starting points. Alternatively, if you've built your own Power BI–based analytical reports by using BYOD or your own data warehouse, you can pin them to analytical workspaces. In both cases, you can enrich business processes through in-context reporting that drives action.
The ready-made analytical workspaces that are part of core Finance and Operations apps (or part of ISV extensions) include reports that are built by using Entity store. Entity store contains aggregate measurements (that is, simplified data structures, such as fact tables and dimensions).

Aggregate measurements will be available in the data lake instead of Entity store. If you're using ready-made analytical workspaces that were built by using Entity store, they will be pointed to Data Lake in an upcoming monthly service update. The Microsoft service team will notify you when your Entity store is ready for the transition to Data Lake. You won't have to modify the reports themselves. Therefore, no development work will be required.

If you've pinned your own Power BI reports to analytical workspaces, you can transition them to Data Lake on your own schedule.

How you can modernize your existing data warehouse by using Data Lake

Although you can use Data Lake integration to transition from BYOD and gain immediate benefits, Data Lake also lets you do much more.

Data Lake is designed to store large amounts of data: hundreds of terabytes (TB) or more. It takes advantage of Azure Blob storage, which is inexpensive storage technology that many underlying services of Azure use.

Data lakes are designed for big data analytics. You can bring your historical data and earlier data from your own systems into Data Lake. This data can consist of scanned documents in addition to business data. You can then apply machine learning models to the documents to make sense of the content and the opinions that the document authors expressed. You can also collect signals from devices and vehicles or machines on the shop floor, and store the data in data lakes. You can then apply machine learning models to detect anomalies and patterns in the signals, and you can join the results with business data to take proactive action.

Data Lake has many associated services that enable analytics, data transformation, and the application of AI and machine learning. Azure services from Microsoft, partners, and open-source tools can be used to reason over the data.

Instead of downloading Finance and Operations data from a data lake into your on-premises data warehouses, you can bring your on-premises data into a data lake. Microsoft refers to this transformation as modernization of data warehouses.

Planning the transition

You can plan your transition to Data Lake in multiple stages, as shown in the following illustration. Each stage offers business benefits that can be justified on their own. You can use the stages that are shown here as a planning guideline.
1. **Your current situation:** You might already be using BYOD and analytical workspaces that are based on Entity store.

2. **Getting easy access to data:** As you gain access to tables, entities, and aggregate measurements in Data Lake, you will be able to retire BYOD and use the data that is readily available. Therefore, management effort and costs can be reduced, as was discussed earlier. You can keep your existing downstream data warehouses and pipelines to manage project scope and budget.

3. **Empowering power users:** Analytical workspaces will be transitioned to Data Lake as a service update. Therefore, ready-made analytical workspaces are based on the same data that is available in the data lake. Power users can easily extend analytical workspaces. When the service update occurs, you will be able to make the full capabilities of PowerBI.com available to your power users. By using capabilities such as Power BI dataflows, power users can easily combine data from online services and data is already available in the data lake. The same reports that are available in analytical workspaces can be consumed directly on PowerBI.com.

4. **Modernizing your previous data warehouse:** Modernization will probably be the investment that brings the most benefits. You can move data that currently exists in your on-premises data warehouse to the cloud. You can rely on cloud-based computing services and apply the same transformations on a pay-per-use basis. You can also combine your business data with sensor and device data.
NOTE
The Export to Data Lake feature is in public preview in the United States, Canada, United Kingdom, Europe, South East Asia, East Asia, Australia, and Japan regions. If your Finance and Operations environment is in any of those regions, you can enable this feature in your environment by using Microsoft Dynamics Lifecycle Services (LCS).

The feature may be temporarily unavailable during the preview period in your environment, or the feature may not be available in your region.

In the coming months, Microsoft will enable this feature in additional regions, and also in additional environments. If your environment isn't in one of the previously listed regions, complete the survey and let us know. You can also join the preview Yammer group. You can use the Yammer group to stay in contact and ask questions that will help you understand the feature.

The Export to Data Lake feature isn't available in Tier-1 (developer) environments. You must have a cloud-based Tier-2 or higher sandbox environment to enable this feature.

In your Tier-1 (developer) environment, you can prototype or plan the feature implementation by using GitHub tools. The tools let you export data from your Tier-1 or sandbox environment into a storage account in the same format that is exported by the feature.

Use of this feature in production environments isn't supported while it's in preview. You can't enable this feature in production environments. You can preview it only in your sandbox (Tier-2 or above) environments.

Create Service Principal for Microsoft Dynamics ERP Microservices

The Export to Azure Data Lake feature is built using a microservice that exports Finance and Operations app data to Azure Data Lake and keeps the data fresh. Microservice uses the Azure service principal, Microsoft Dynamics ERP Microservices, to securely connect to your Azure resources. Before you configure the Export to Data Lake feature, add the Microsoft Dynamics ERP Microservices service principal to your Azure Active Directory (Azure AD). This step enables Azure AD to authenticate the microservice.

NOTE
You will need Azure Active Directory global administrator rights to perform these steps.

To add the service principal, complete the following steps.

1. Launch the Azure portal and go to the Azure Active Directory.

2. On the left menu, select Manage > Enterprise Applications, and search for the following applications.

<table>
<thead>
<tr>
<th>APPLICATION</th>
<th>APP ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Dynamics ERP Microservices</td>
<td>0cdb527f-a8d1-4bf8-9436-b352c68682b2</td>
</tr>
</tbody>
</table>

If you are unable to find the above applications, complete following steps.

3. On your local machine, open the Start menu, and search for PowerShell.

4. Right-click Windows PowerShell, and then select Run as administrator.

5. Run the following command to install AzureAD module:
Configure Azure Resources

To configure the export to Data Lake, create a storage account in your own Azure subscription. This storage account is used to store data. Next, create an Azure AD application ID that grants access to the root of your storage account. Your Finance or Operations app will use the Azure AD application to gain access to storage, create the folder structure, and write data. Create a key vault in your subscription and store the name of the storage account, application ID, and the application secrets. If you don't have permission to create resources in Azure portal, you will need assistance from someone in your organization with the required permissions.

The steps, which take place in the Azure portal, are as follows:

1. Create an application in Azure Active Directory
2. Create a Data Lake Storage (Gen2 account) in your subscription
3. Grant access control roles to applications
4. Create a key vault
5. Add secrets to the key vault
6. Authorize the application to read secrets in the key vault
7. Power Platform integration
8. Install the Export to Data Lake add-in in LCS

Create an application in Azure Active Directory

1. In the Azure portal, select Azure Active Directory, and then select App registrations.
2. Select New registration, and enter the following information:
   - Name: Enter a name for the app.
   - Supported Account types: Choose the appropriate option.
3. After the application is created, select it, and then copy and save the Application (client) ID at the top of the page. You will need this later.
4. On the left navigation pane, select API permissions.
5. Select Add a permission, and in the Request API permissions dialog box, select Azure Key vault.
6. Select Delegated permissions, select user_impersonation, and then select Add permissions.
Create a Data Lake Storage (Gen2) account in your subscription

The Data Lake Storage account will be used to store data from your Finance and Operations apps. To manually create a storage account, you must have administrative rights to your organization's Azure subscription. To create a storage account, complete the following steps.

1. In the Azure portal, select Create new resource, and then search for and select Storage account – blob, file, table, queue.

2. In the Create storage account dialog box, provide values for the following parameter fields:
   - **Location:** Select the data center where your environment is located. If the data center that you select is in a different Azure region, you may incur additional data movement costs. If your Microsoft Power BI or your data warehouse is in a different region, you can use replication to move storage between regions.
   - **Performance:** We recommend you select Standard.
   - **Account kind:** You must select StorageV2. In the Advanced options dialog box, you will see the option, Data Lake storage Gen2.

3. On the Advanced tab, select Data Lake storage Gen2 > Hierarchical namespaces, and then select Enabled. If you disable this option, you may not be able to consume data that is written by Finance and Operations apps with services such as Power BI data flows and AI builder.

4. Select Review and create. When the deployment is complete, the new resource will be shown in the Azure portal.

5. In the Azure portal, select the storage account you created. Copy and save the storage account name.

Grant access control roles to applications

You need to grant your application permissions to read and write to the storage account. These permissions are granted by using Roles in Azure AD.

1. In Azure portal, open the storage account that you created earlier.

2. Select Access Control (IAM) in the left navigation.

3. On the Access control page, select the Role assignments tab.

4. Select Add at the top of the page, and then select Add role assignment.

5. In the Add role assignment dialog box, select the Role field, and then select Storage blob data contributor.

6. In the Select field, select the application that you registered earlier.

**NOTE**

Don't make any changes to the fields, Assign access to and Azure AD user, group, or service principal.
7. Select **Save**.

8. Repeat steps 4-7 to add the **Storage blob data reader** role, as shown.

9. Validate the storage account role assignment for the application you created earlier.

<table>
<thead>
<tr>
<th>APPLICATION</th>
<th>ROLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>The application you created earlier</td>
<td>Storage blob data contributor</td>
</tr>
<tr>
<td>The application you created earlier</td>
<td>Storage blob data reader</td>
</tr>
</tbody>
</table>

### Create a key vault

A key vault is a secure way to share details such as storage account name to your Finance and Operations apps. Complete the following steps to create a key vault and a secret.

1. In the Azure portal, select **Create a new resource** and then search for and select **Key Vault**.
2. In the **Create key vault** dialog box, in the **Location** field, select the datacenter where your environment is located.
3. After the key vault is created, select it from the list, and on the left navigation pane, select **Overview**.
4. Save the value in the **DNS name** field. You will need this value later.

### Add secrets to the key vault

You are going to create three secrets in the Key vault and then add the values saved from previous steps. For each of the secrets, you will need to provide a secret name and provide the value you saved from earlier steps.

<table>
<thead>
<tr>
<th>SUGGESTED SECRET NAME</th>
<th>SECRET VALUE THAT YOU SAVED EARLIER</th>
<th>EXAMPLE SECRET VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>app-id</td>
<td>The ID of the application created earlier.</td>
<td>8936e905-197b-xxxx-xxxx-xxxxxxxxx</td>
</tr>
<tr>
<td>app-secret</td>
<td>The client secret specified earlier.</td>
<td>NaeIxxxxxxx---xxxx7eixx~1g-</td>
</tr>
<tr>
<td>storage-account-name</td>
<td>The name of the storage account created earlier.</td>
<td>contosod365datalake</td>
</tr>
</tbody>
</table>

You will need to complete the following steps three times, once for each secret.

1. In the Azure portal, go to the key vault you created earlier and on the left navigation pane, select **Secrets**.
2. Select **Generate/Import**, and in the **Create a secret** dialog box, in the **Upload options** field, select **Manual**.
3. Enter a name for the secret. See the table in the introduction of this section for suggested names.
4. Copy and paste the corresponding secret value in the **Value** field.
5. Select **Enabled**, and then select **Create**.

You will notice the secret created in the list of secrets.

### Authorize the application to read secrets in the key vault

1. In **Azure portal**, open the key vault that you created earlier.
2. In the **Add access policy** dialog box, select **Add**.
3. On the left navigation pane, select **Access policies** > **Add Access Policy** to create a new policy.
4. In the **Add access policy** dialog box, in the **Select principal** field, locate and select the application, *Microsoft Dynamics ERP Microservices*, and then click **Select**.

   **NOTE**
   
   If you can’t find *Microsoft Dynamics ERP Microservices*, see the Create Service Principal section in this document.

5. In the **Secret permissions** fields, select **Get** and **List**.

6. In the **Access policy** dialog, select **Add**.

   You should see application with access to your key vault as shown below.

<table>
<thead>
<tr>
<th>APPLICATION</th>
<th>SECRET PERMISSIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Dynamics ERP Microservices</td>
<td>Get, List</td>
</tr>
</tbody>
</table>

7. Select **Save**.

**Power Platform integration**

If this is the first time you are installing add-ins in this environment, you may need to enable the **Power Platform integration** for this environment. There are two options to set up Power Platform integration in Finance and Operations app environments.

**Option 1: Set up Power Platform integration using LCS**

To set up Power Platform integration from LCS, see Add-ins overview.

**Option 2: Set up Power Platform integration using the Dual-write wizard**

Another way to set up **Power Platform integration** is to create a Power Platform environment with a database and then use the Dual-write setup. Complete the following steps to create the Power Platform environment and complete the integration.

1. Create an environment with database.
2. Complete the requirement and prerequisite.
3. Use the dual-write wizard to link your environment.
4. Validate that the Power Platform integration is set up and added in the LCS environment page.

   **NOTE**
   
   If you use this approach, you must select a Power Platform environment that is in the same region as your Finance and Operations environment. If you select a Power Platform environment that is in a different region, installation of the add-in might fail.

**Install the Export to Data Lake add-in in LCS**

Before you can export data to your data lake from your Finance and Operations apps, you must install the **Export to Data Lake** add-in in LCS. To complete this task, you must be an environment administrator in LCS for the environment that you want to use.

You need the following information before you start. Keep the information handy before you begin.
<table>
<thead>
<tr>
<th>INFORMATION YOU NEED FOR EXPORT TO DATA LAKE ADD-IN</th>
<th>WHERE YOU CAN FIND IT</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Your environment Azure AD Tenant ID</td>
<td>Your Azure AD tenant ID in the Azure portal. Sign in to the Azure portal and open the Azure Active Directory service. Open the Properties page and copy the value in the Directory ID field.</td>
<td>72f988bf-0000-0000-0000-2d7cd011db47</td>
</tr>
<tr>
<td>DNS name of your key vault</td>
<td>This name should have been previously saved. Enter the DNS name of your key vault</td>
<td><a href="https://contosod365datafeedpoc.vault.azure.net">https://contosod365datafeedpoc.vault.azure.net</a></td>
</tr>
<tr>
<td>The secret that contains the name of your storage account</td>
<td>If you used the suggested name, enter storage-account-name. If not, enter the secret name you defined.</td>
<td>storage-account-name</td>
</tr>
<tr>
<td>Secret that contains the Application ID</td>
<td>If you used the suggested name, enter app-id. If not, enter the secret name you defined.</td>
<td>app-id</td>
</tr>
<tr>
<td>Secret that contains the Application secret</td>
<td>If you used the suggested name, enter app-secret. If not, enter the secret name you defined.</td>
<td>app-secret</td>
</tr>
</tbody>
</table>

1. Sign in to LCS and navigate to your environment.
2. On the Environment page, select the Environment add-ins tab. If Export Data Lake appears in the list, the Data Lake add-in is already installed, and you can skip the rest of this procedure. Otherwise, complete the remaining steps.
3. Select Install a new add-in, and in the dialog box, select Export to Data lake. If Export to data lake isn't listed, the feature might not be available for your environment at this time.
4. In the Setup add-in dialog box, enter the required information. To answer the questions, you must already have a storage account. If you don’t already have a storage account, create one, or ask your admin to create one on your behalf.
5. Accept the terms of the offer by selecting the check box, and then select Install.

The system installs and configures the data lake for the environment. This operation might take a few minutes. After installation and configuration are completed, Export to Data Lake should be listed on the Environment page, and the status should be Installed. If a different status is shown, see the "Troubleshooting" section that follows.

**Troubleshooting**

**Add-in installation isn't completed within a few minutes**

In some cases, add-in installation might show a status of Installing or Configuring for more than 10 minutes. The cause of the delay might be a configuration issue or a missing parameter. In this case, select the Abort option, and then follow the steps to install the add-in again.

**Add-in installation fails**

In some cases, add-in installation might show a status of Installation failed. When installation fails, an error code and error message are shown. The "Resolution" column in the following table provides suggestions that can help you correct the reason for the failure. To correct the issue, select the Abort option, and then follow steps to install the add-in again.
<table>
<thead>
<tr>
<th>ERROR CODE AND MESSAGE</th>
<th>RESOLUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AppidUserError</strong>: Failed to find Application ID to access the data lake. Application ID provided is incorrect or can't be found.</td>
<td>The application ID (<strong>app-id</strong>) that is provided in the key vault can't be found in Azure AD. Validate the application ID by following the steps in <strong>Configure export to Azure Data Lake - Create Application</strong>. You might have to contact the system administrator or the administrator who configured Azure resources.</td>
</tr>
<tr>
<td><strong>AppSecretUserError</strong>: Failed to access data lake with given Application ID and Application secret.</td>
<td>The application ID (<strong>app-id</strong>) and application secret (<strong>app-secret</strong>) that are provided can't be used to access the storage account. Validate the application ID and application secret by following the steps in <strong>Configure export to Azure Data Lake - Create Application</strong>. Next, verify that the application has the required access to the storage account. For more information, see <strong>Configure export to Azure Data Lake - Grant access</strong>. You might have to contact the system administrator or the administrator who configured Azure resources.</td>
</tr>
<tr>
<td><strong>StorageNameUserError</strong>: Failed to access the storage account using the storage name provided in the key vault.</td>
<td>The storage account that is provided in the key vault can't be found, or it isn't valid. Verify that the correct storage account name is entered in the key vault by following the steps in <strong>Configure export to Azure Data Lake - Add secrets</strong>. Verify that you've provided the correct secret name for the storage account by following the steps in <strong>Configure export to Azure Data Lake Add secrets</strong>.</td>
</tr>
<tr>
<td><strong>KeyVaultUserError</strong>: Failed to access the key vault or the key vault secrets.</td>
<td>The service can't access the key vault or the secrets in it. Verify that your Azure subscription hasn't expired. Verify that you've created the service principal by following the steps in <strong>Configure export to Azure Data Lake - Create service principal</strong>. Verify that the key vault contains all the required secrets by following the steps in <strong>Configure export to Azure Data Lake - Add secrets</strong>. Verify that you've provided the correct key vault URI in the configuration steps in <strong>Configure export to Azure Data Lake - Install add-in</strong>.</td>
</tr>
<tr>
<td><strong>TenantIdUserError</strong>: Failed to locate the Azure Tenant ID for the environment.</td>
<td>Verify that you've provided the correct Azure tenant ID by following the steps in <strong>Configure export to Azure Data Lake - Install add-in</strong>.</td>
</tr>
</tbody>
</table>
The Export to Data Lake feature is in public preview in the United States, Canada, United Kingdom, Europe, South East Asia, East Asia, Australia, and Japan regions. If your Finance and Operations environment is in any of those regions, you can enable this feature in your environment by using Microsoft Dynamics Lifecycle Services (LCS).

The feature may be temporarily unavailable during the preview period in your environment or the feature may not be available in your region.

In the coming months, Microsoft will enable this feature in additional regions, as well as in additional environments. If your environment isn’t in a region where the preview is enabled, complete the survey and let us know. You can also join the preview Yammer group. You can use the Yammer group to stay in contact and ask questions that will help you understand the feature.

The Export to Data Lake feature lets you copy data from your Finance and Operations apps into your own data lake (Azure Data Lake Storage Gen2). The system lets you select the tables and entities that are included. After you select the data that you want, the system makes an initial copy. The system then keeps the selected data up to date by applying changes, deletions, and additions. After data changes in your Finance and Operations app instances, there might be a delay of a few minutes before the data is available in your data lake.

Turn on the Export to Data Lake feature

Before you can use this feature, see Configure export to Azure Data Lake.

Select data

You can select the tables and entities that should be staged in Data Lake.

1. In your environment, go to System Administration > Setup > Export to Data Lake.

   You can also open the Export to Data Lake page by using the search field on the navigation bar. Enter Configure in the search field. The search results should include a link to the page.

2. On the Export to Data Lake page, on the Choose Tables tab, select the data tables that should be staged in Data Lake. You can search for tables by either display name or system name. You can also see whether a table is being synced.

3. When you’ve finished, select Add Tables to add the selected tables to Data Lake.
4. Select **Activate data feed**, and then select **OK**. When you add a table, the system might show its status as **Initializing**. This status indicates that the system is making an initial copy of data. When the initial copy is completed, the system changes the status to **Running**.

In the event of an error, the system shows the status as **Deactivated**.

You can consume data in the data lake when the status is **Running**. If you consume data in the data lake while the status is **Initializing** or **Deactivated** status, you might not see all the data.

If you aren't familiar with the specific tables that you require, you can select tables by using entities. Entities are a higher-level abstraction of data and might include multiple tables. By selecting entities, you also select the tables that include them.

**NOTE**

When you open the **Choose using Entities** tab for the first time, you might notice that the list of entities on the page is empty. The system might require some time to fill in the list of entities. You can force a refresh of the list by selecting **Manage > Rebuild data feed catalog** on the Action Pane.

5. On the **Choose using Entities** tab, select the entities, and then select **Add Tables using Entities**.
Monitor the tables in Data Lake

You don’t have to monitor or schedule data exports, because the system keeps the data updated in Data Lake. However, you can view the status of ongoing data exports in the Status column on the Export to Data Lake page.

Troubleshooting common issues and errors

**Export to Data Lake feature is not available in your region and/or your environment at this time**

This feature is not available in Tier-1 (developer) environments. You need a sandbox environment (Tier 2 or higher) with Platform updates for version 10.0.13 or higher.

This feature is in limited preview and may not be available in all Azure regions where Finance and Operations apps are available, or this feature may not be available for your environment. If you would like to join a future preview, complete the survey. We will contact you when we are ready to include you. You can also join a Yammer group by completing the survey. You can use the Yammer group to stay in contact and ask questions that will help you understand the feature. We are working hard to make this feature available soon.

**Export to Data Lake feature is currently being installed for your environment. Please check back later.**

Before you can use this feature, you need to configure the export to Data Lake. For more information, see Configure export to Azure Data Lake.

**Export to Data Lake add-in is not installed.**

Ask your administrator to install this add-in using Dynamics Lifecycle Services (LCS). Before you can use this feature, you need to configure the export to Data Lake. For more information, see Configure export to Azure Data Lake.

**Export to Data Lake feature failed to install in Dynamics Life Cycle Services (LCS).**

Ask your administrator to re-install the Export to Data Lake add-in. If this issue persists, contact Support. When you configure the Export to Data Lake feature, the system may report an error. Or, there may be an error when you access the data lake after configuration due to a change in your environments. For more information, see Configure export to Azure Data Lake.

**Export to Data Lake feature is temporarily unavailable. Please check back later.**

If you see this error for a prolonged period of time, contact Support.

**Status codes with extended errors**

When an error occurs in a table that you added to Export to Data Lake, you may see an error code in the status column. The following error codes provide the cause of the error and how to correct the issue.

<table>
<thead>
<tr>
<th>ERROR CODE</th>
<th>ISSUE</th>
<th>NEXT STEPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>The table you added doesn't contain a RecID field.</td>
<td>RecID fields are used by the system to index table data. Tables that don't contain a RecID field can't be added to the Data Lake. If this issue is from a table in a Finance and Operations app, contact Microsoft support. If this table was developed by your partner or ISV, contact the developer to include a RecID field.</td>
</tr>
<tr>
<td>ERROR CODE</td>
<td>ISSUE</td>
<td>NEXT STEPS</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>------------</td>
</tr>
<tr>
<td>401</td>
<td>The table you added is missing in the database.</td>
<td>The table you added is no longer available in the database and the system can’t continue updating data in the lake. The table may have been removed because of a software or database update. To resolve this issue, contact your database administrator or a developer. If this table was developed by your partner or ISV, contact the developer. You may also encounter this issue when you add “derived tables” such as the DirPerson table. Derived tables are currently not supported by the service. To choose derived tables, you need to choose the base table. There are plans to add support for derived tables in a future release.</td>
</tr>
<tr>
<td>402</td>
<td>The RecID field isn’t indexed.</td>
<td>The system has detected that the RecID field contained in the table isn’t part of an index or is not the first field in an index. This may lead to poor performance in updating the data lake and updates may take longer to reflect in the data lake. You can include the RecID field in an index to resolve this issue. If this issue is from a table in a Finance and Operations app, contact Microsoft support. If this table was developed by your partner or ISV, contact the developer.</td>
</tr>
<tr>
<td>409</td>
<td>Failed to access the storage account due to permissions. Verify the storage account configuration.</td>
<td>The system is unable to access and write to the storage account. Ensure that you have enabled Hierarchical Name Spaces (HNS) and validate access roles. For more information, see Configure export to Azure Data Lake - Grant access. To resolve this issue, contact the system administrator or the LCS administrator who configured the system.</td>
</tr>
<tr>
<td>410</td>
<td>Failed to find Application ID to access the data lake. Application ID provided is incorrect or can’t be found.</td>
<td>The Application ID provided in the key vault can’t be found in Azure Active Directory. Validate the Application ID provided in the key vault by following the steps in Configure export to Azure Data Lake - Create application.</td>
</tr>
<tr>
<td>ERROR CODE</td>
<td>ISSUE</td>
<td>NEXT STEPS</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>------------</td>
</tr>
<tr>
<td>411</td>
<td>Failed to access data lake with given Application ID and Application secret.</td>
<td>Application ID (app-id) and Application secret (app-secret) provided can't be used to access the storage account. Verify the Application ID and Application secret are valid by following the steps in Configure export to Azure Data Lake - Create Application. Next, validate that the application has required access to the storage account. For more information, see Configure export to Azure Data Lake - Grant access. You need to contact the systems administrator or the LCS administrator who configured the system.</td>
</tr>
<tr>
<td>415</td>
<td>Failed to access the storage account using the storage name provided in the key vault.</td>
<td>Storage account provided in the key vault can't be found or is invalid. Validate that the correct storage account name is entered into the key vault by following the steps in Configure export to Azure Data Lake - Add secrets. Verify that you have provided the correct secret name for the storage account by following the steps in Configure export to Azure Data Lake Add secrets.</td>
</tr>
<tr>
<td>420</td>
<td>Failed to access the key vault or the key vault secrets.</td>
<td>Service can't access the key vault or the secrets in the key vault. Verify that your Azure subscription has not expired. Verify that you have created the service principal by following the steps in Configure export to Azure Data Lake - Create service principal. Verify that the Key vault contains all the required secrets by following the steps in Configure export to Azure Data Lake - Add secrets. Verify that you have provided the correct key vault URI in the configuration steps in Configure export to Azure Data Lake - Install add-in.</td>
</tr>
<tr>
<td>425</td>
<td>Failed to locate the Azure Tenant ID for the environment.</td>
<td>Verify that you have provided the correct Azure tenant ID by following the steps in Configure export to Azure Data Lake - Install add-in.</td>
</tr>
<tr>
<td>ERROR CODE</td>
<td>ISSUE</td>
<td>NEXT STEPS</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>------------</td>
</tr>
<tr>
<td>430</td>
<td>Failed to access the environment.</td>
<td>Ensure that the environment is available and not in a deleted or inactive state.</td>
</tr>
<tr>
<td>435</td>
<td>Files in the lake are corrupted or invalid.</td>
<td>The system has detected corrupted files or an invalid folder structure in your data lake. The system manages files and folders in the lake provided in the configuration. You should not modify files or the folder structure in the data lake yourself. Verify that your users or a process has not modified files or folders in the lake. You can reactivate the tables to see if the issue is resolved. If this does not address the issue, uninstall and re-install the Export to Data Lake add-in.</td>
</tr>
</tbody>
</table>

**Error status codes 5xx indicate a system error encountered while exporting data**

Due to the error, the system has paused data export – data that exists in the lake won't be updated until the error is resolved. Try to deactivate and activate the table to see if this resolves the issue. Note that deactivating and activating the table may cause the system to re-initialize the data in the lake by taking a full copy. If the issue persists, contact Microsoft support with the table name and the error code.

**Error status codes 8xx and 9xx indicate a change to the table or database structure since it was added**

<table>
<thead>
<tr>
<th>ERROR CODE</th>
<th>ISSUE</th>
<th>NEXT STEPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>800</td>
<td>There is a change in the data table schema.</td>
<td>The system found a change in the table structure when adding changed data to the lake. This change is typically the result of a software update or a database modification. For example, when a new field is added to a table, or a field is modified or removed, the data updated in the lake may not be in the same format as the existing data. Because of the error, the system has paused data export. The data that exists in the lake won't be updated until the error is resolved. Deactivate and activate the table to see if the issue is resolved.</td>
</tr>
<tr>
<td>801</td>
<td>There is an issue with the Change data capture feature.</td>
<td>The Export to Data Lake feature uses the Change data capture feature. Change data can't be exported because the Change data capture feature is disabled in the Finance and Operations apps database. This may be the result of a database maintenance operation. Deactivate and activate the table to see if the issue is resolved. If the issue persists, contact Microsoft support.</td>
</tr>
<tr>
<td>ERROR CODE</td>
<td>ISSUE</td>
<td>NEXT STEPS</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>------------</td>
</tr>
<tr>
<td>802</td>
<td>There is an issue with the Change data capture feature for the table.</td>
<td>The Export to Data Lake feature uses the Change data capture feature. Change data can't be exported because the Change data capture feature isn't enabled for this table in your Finance and Operations apps database. This may be the result of a database maintenance operation. Deactivate and activate the table to see if the issue is resolved.</td>
</tr>
<tr>
<td>900</td>
<td>There is an issue with the Change data capture feature in this environment.</td>
<td>The Export to Data Lake feature uses the Change data capture feature. Change data can't be exported because the Change data capture feature isn't enabled for this table in the Finance and Operations apps database. This may be the result of a database maintenance operation. Deactivate and activate the table to see if the issue is resolved. If the issue persists, contact Microsoft support.</td>
</tr>
</tbody>
</table>

**NOTE**

Deactivating and activating the table may cause the system to re-initialize the data in the lake by creating a full copy. If this is a large table, the initialize process may take some time. In the future, the system may automatically update data in the lake to reflect the table structure changes.
IMPORTANT

This feature is currently in public preview. This feature is comprised of the following components:

- Extend analytical workspaces by using PowerBI.com - Available in a future platform update.

Automated Entity store refresh

You need to enable automated Entity store refresh before enabling Data Lake integration.

1. Go to **System administration** > **Set up** > **Entity store**.
   
   On the **Entity store** page, a message indicates that you can switch to the **Automated Entity store refresh** option. This option is managed by the system. An admin doesn’t have to schedule or monitor the Entity store refresh.

2. Select **Switch now**.

   **IMPORTANT**
   
   This action isn’t reversible. After you switch to the **Automated Entity store refresh** option, you can’t revert to the old user interface (UI) experience.

3. Select **Yes** to continue.

You will now see the new experience.
After the new experience is turned on, you can define the refresh for each aggregate measurement. The following refresh options are available:

- Every hour
- Twice a day
- Once a day
- Once a week

In addition, an admin can refresh any aggregate measurement on demand by selecting the Refresh button. Additional options will be added in future platform updates. These options will include options for real-time refresh.

**IMPORTANT**

When the automated refresh is enabled, in some cases the system may disable refresh of Aggregate measurements. You must revisit aggregate measurements and validate that appropriate refresh intervals have been applied by the system.

When this feature is turned on, Entity store data isn’t populated in the relational Entity store database in the Microsoft subscription. Instead, it’s populated in an Azure Data Lake Storage Gen2 account in your own subscription. You can use the full capabilities of PowerBI.com and other Azure tools to work with Entity store.

Before you start, you must complete these tasks in the Azure portal.

1. **Create storage accounts.** Provision a storage account in the same data center where your environment is provisioned. Make a note of the connection string for the storage account, because you will have to provide it later.
2. **Create a Key Vault and a secret.** Provision Azure Key Vault in your own subscription. You will need the
Create storage accounts
1. In the Azure portal, create a new storage account.
2. In the Create storage account dialog box, provide values for the following parameter fields:
   - Location: Select the data center where your environment is located. If the data center that you select is in a different Azure region, you will incur additional data movement costs. If your Microsoft Power BI and/or your data warehouse is in a different region, you can use replication to move storage between regions.
   - Performance: We recommend that you select Standard.
   - Account kind: You must select StorageV2.
3. In the Advanced options dialog box, you will see the Data Lake storage Gen2 option. Select Enable under the Hierarchical namespaces feature. If you disable this option, you can’t consume data written by Finance and Operations apps with services such as Power BI data flows.
4. Select Review and create. When the deployment is completed, the new resource will be shown in the Azure portal.
5. Select the resource, and then select Settings > Access keys.
6. Make a note of the connection string value, because you will have to provide it later.

Create a Key Vault and a secret
1. In the Azure portal, create a new Key Vault.
2. In the Create key vault dialog box, in the Location field, select the data center where your environment is located.
3. After Key Vault is created, select it in the list, and then select Secrets.
4. Select Generate/Import.
5. In the Create a secret dialog box, in the Upload options field, select Manual.
6. Enter a name for the secret. Make a note of the name, because you will have to provide it later.
7. In the value field, enter the connection string that you obtained from the storage account in the previous procedure.
8. Select Enabled, and then select Create. The secret is created and added to Key Vault.

Register the app
1. In the Azure portal, select Azure Active Directory, and then select App registrations.
2. Select New registration at the top of the menu, and enter the following information:
   - Name - Enter a friendly name for the app.
   - Select Accounts in this Organizational directory only unless your storage account and your Dynamics environment are in different Azure Active Directory domains.
3. After the application is created, select API permissions.
4. In the dialog box that appears, select **Add a permission**.

5. You will see a dialog box with a list of APIs. In the list, select **Azure Key Vault**.

6. Select the **Delegated permissions** box, select **user_impersonation**, and then select **Add permissions** to save your changes.

7. Select the **Certificates & secrets** menu on the left navigation pane, and then select **New client secret**.

8. In the **Description** field, enter a name and choose an expiry period. Select **Add**.

9. A secret is generated and shown in the **Value** field.

10. Immediately copy the secret to the clipboard, because it will disappear within one or two minutes. You will have to provide this key to the application later.

**Add a service principal to Key Vault**

1. In the Azure portal, open Key Vault that you created earlier.

2. Select **Access policies**, and then select **Add** to create a new access policy.

3. In the **Select principal** field, select the name of the application that you previously registered.

4. In the **Key permissions** field, select **Get** and **List** permissions.

5. In the **Secret permissions** field, select **Get** and **List** permissions.
Work in Entity store in a Data Lake

1. Go to System administration > Set up > System parameters.

2. On the Data connections tab, enter the following information that you made a note of earlier in this topic:
   - **Application ID**: Enter the application ID of the Azure AD application that you registered earlier.
   - **Application Secret**: Enter the application key (secret) for the Azure AD application.
   - **DNS name**: Enter the DNS name of Key Vault.
   - **Secret name**: Enter the name of the secret that you added to Key Vault together with connection string information.

6. Select **Save**.
3. Select the Test Azure Key Vault and Test Azure Storage links to validate that system can access the configuration information that you provided.

4. Select the Enable data connection check box.

Entity store data should now be populated in the storage location that you provided, not in the relational Entity store database.

The aggregate measurements and refresh options that you select in the Entity store UI should now apply to data that is copied to Data Lake.
Change data in a data lake lets you build near-real-time data pipelines that react to data changes in Finance and Operations apps. The Change feed folder in the data lake contains every data change in Finance and Operations apps. This folder is automatically created by the Export to Data Lake feature.

Why do you need change data in a data lake?

Data in a data lake is often used for reporting purposes. Although you can use the table data in the data lake to create reports, you can also create additional copies of the data to improve your reporting. For example, you might have a data mart that is designed to enable your power users. In this data mart, you might have simplified, often aggregated, fact tables and dimension tables.

As table data in the data lake is updated, you must keep the corresponding fact tables and dimension tables in the data lake updated. Otherwise, your reports won't reflect the latest data.

The easiest way to update fact tables and dimension tables is to periodically create a full copy by using tables. However, this approach can be inefficient. If your tables are large (for example, if they have tens of millions or hundreds of millions of rows), the process of updating a fact table by making a full copy might take hours and consume lots of compute resources. Therefore, your users might not have the reports in time (that is, they might have to wait hours to see the latest data in reports). Additionally, because compute resources are consumed every time that data is reprocessed, you might receive a larger bill from the services that you've consumed.

Incremental update of your fact tables and dimension tables provides the answer to both problems (time consumption and compute resource consumption). In an incremental update, you select only the changed records from source tables, and update them in corresponding fact tables and dimension tables.

Incremental update is a standard capability in most data transformation tools, such as Azure Data Factory. However, for the incremental update feature to work, you must identify the records that changed in source tables.

The Change feed folder provides a history of table data changes in the data lake. This history can be used for data pipelines that use incremental update.

The Change feed folder

The Change feed feature relies on a SQL Server feature that is named Change Data Capture (CDC). CDC is the native approach to capturing change data in a SQL Server database, which is the data store behind Finance and Operations apps. The Change feed feature lets you access the CDC change log in your data lake.

The following illustration shows how change feeds work in Finance and Operations apps.
1. Whenever a data change occurs in Finance and Operations apps, the underlying database (AXDB) is updated. The CDC feature ensures that the update is reflected in the database. CDC captures the changes in a log (the change log), together with a logical sequence number (LSN value), a date/time stamp (Change Date-Time value), and a Change Payload value that identifies the data that changed.

2. **Export to Data Lake** microservices capture the changes in the database and write the change log to the customer’s data lake. Change feed folders in the data lake contain the change log, which is organized into folders.

3. In addition, in the **Tables** folder, each row that changed also contains several new fields. Each row contains the LSN value of the corresponding change record and the Change Date-Time value. Although you can use the LSN and Change Date-Time fields in the table folders to identify whether a row changed, they contain only the latest change. If the same row changed multiple times, only the latest change is shown in the Tables folder.

A given change feed folder in the data lake and the changed fields in the corresponding table folders are consistent. Therefore, a single microservice updates both the change feed folder and the corresponding table in the data lake at the same time.

### Exploring the Change feed folder in your data lake

Change feeds are automatically enabled when you add tables to a data lake.

When you add a table to a data lake, or when you activate a table that has been inactivated, the system makes an initial copy of the data in the data lake. At this point, the table’s status is shown as **Initializing**. When the initial copy is completed, the system changes the status to **Running**. When the table is in **Running** status, changes in the Finance and Operations database are reflected in the data lake, and change feeds are added.

To access change folders, open the Azure portal, and find and select the storage account that is associated with your Finance and Operations environment. You should see the **Change feed** folder in the data lake folder structure. The following illustration shows an example.
When you open the Change feed folder, you should see folders that correspond to the tables that you've added to the data lake. You should also see CDM metadata files that describe the change folder data. The following illustration shows an example.

CDM metadata files describe the structure of change feed data that is contained in folders. You can use the CDM metadata files and data transformation tools such as Data Factory to read change feed data without having to read raw comma-separated values (CSV) files. To examine the metadata, select a metadata file, and then open it in a text editor.
As you should notice from the metadata definitions, the Change feed folder contains the CDC change log details, together with additional fields. The following illustration and table provide details about the format of changes in change folders.

**Start_LSN**
This field identifies the LSN of the transaction that changed source data in the Finance and Operations database.

**Note:** The Start_LSN value is not enclosed in double quotation marks in CSV files. It’s a hexadecimal value, as represented in the SQL Server database. Here is a sample value: 0X00011E9F00000FB00001.

**End_LSN**
This field isn't used.

**DML_Action**
Each change is stored as a separate record. The DML_Action field identifies the change that was made to the record.

- 1: DELETE
- 2: INSERT
- 3: BEFORE_UPDATE
- 4: AFTER_UPDATE

**Note:** The system doesn't add a BEFORE_UPDATE record to change feeds.
<table>
<thead>
<tr>
<th>FIELD NAME</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq_Val</td>
<td>This field identifies the sequence number within the LSN that changed data in the source. Because a transaction might update more than one table in the Finance and Operations database, the Seq_Val field indicates the sequence number that CDC assigned to the table. A change record is added for every change that is made to a table in a transaction. If the same record was updated multiple times in a single transaction, you will find multiple change records that have a separate sequence number. In the future, in extreme cases (for example, when there are thousands of updates to the same record in a single transaction), the system might store the latest update record.</td>
</tr>
<tr>
<td>Update_Mask</td>
<td>A bitmap that identifies the fields that changed. This bitmap resembles the update mask in Change tracking. However, by examining the bitmap, you can identify the fields that changed.</td>
</tr>
<tr>
<td>List of fields and values</td>
<td>The remaining columns provide a list of fields that are present in the table, together with the values. You should use the update mask to identify fields that changed as part of the transaction.</td>
</tr>
<tr>
<td>LastProcessedChange_DateTime</td>
<td>This field provides the value of the CDC Change DateTime field from the Finance and Operations database. The date/time is expressed in Coordinated Universal Time (UTC), per ISO 8601.</td>
</tr>
<tr>
<td></td>
<td>Here is a sample value: &quot;2020-08-24T05:26:03.8622647Z&quot;. Notice that this value is enclosed in double quotation marks. Additionally, it includes the default seven digits of precision after the second value, and a Z that signifies UTC.</td>
</tr>
<tr>
<td>DataLakeModified_DateTime</td>
<td>This field provides the date and time of writing to the data lake. The date/time is expressed in UTC, per ISO 8601.</td>
</tr>
<tr>
<td></td>
<td>Here is a sample value: &quot;2020-08-24T05:26:03.8622647Z&quot;. Notice that this value is enclosed in double quotation marks. Additionally, it includes the default seven digits of precision after the second value, and a Z that signifies UTC.</td>
</tr>
</tbody>
</table>

**Best practices when change feeds are used**

Change feeds are a powerful feature that is enabled by the Export to Data Lake feature in Finance and Operations apps. This section examines some best practices that you should follow when change feeds are used.

**Updating near-real-time data marts**

If you require that your data warehouse or data marts be updated in near-real time (in other words, if they must be updated within minutes of a data change in Finance and Operations apps), you should use change feeds.

However, there are several important concepts that must be understood:
Simplifying BYOD-based ETL pipelines

- Change records are grouped into files that are around 4 megabytes (MB) or 8 MB in size. Microsoft has optimized file sizes so that they provide optimum query response times when the files are queried by Synapse SQL Serverless. Optimized file sizes (and batched writes) also reduce the Azure charges that you might incur as the data lake is updated. Change records are only appended. In other words, they are never updated. However, files in the Change feed folder might occasionally be updated. For example, smaller records might be filled up so that they match the 4 MB size. You should not rely on the date/time stamp of CSV files to identify changes. Instead, you should rely on the LSN or the time stamps in the change record.

- Change feeds are constantly updated in the data lake. Updates might occur in small batches that update the data every minute. These updates might even occur more than one time per minute in the case of tables that are frequently updated. Although you can observe the Change feed folder (or time stamps of files) to trigger downstream jobs, you should consider the possibility that a file might have existing change records that you consumed in the past.

- Your downstream jobs can be orchestrated on a periodic basis, and they can be triggered as microbatches instead of observing changes to files or folders.

- Your downstream data pipelines must have a “last processed” marker (also known as a watermark). Whenever you can, you should rely on the LSN as the watermark. However, you can also use the Change Date-Time value as the watermark. By relying on the LSN, you will ensure that you consume the changes in the same sequence in which they were committed in the Finance and Operations database.

- When you reactivate a table in Finance and Operations, the Change feed folder is cleared, and the system starts change feeds from the next available change. This behavior ensures that the changes are consistent with the Tables folder. When tables are reactivated, you should consider triggering a full refresh of the downstream data pipeline.

A sample Synapse template is available. You can use it to incrementally ingest data into a SQL-based data warehouse.

**Simplifying BYOD-based ETL pipelines**

If you're currently using the bring your own database (BYOD) feature, you can rely on exports of entities that are based on Data management framework (DMF) system tables or batch tables. You might be using export job execution data in DMF system tables to identify the time periods of export jobs. Your downstream jobs might be triggered through job execution status and details that are obtained from DMF tables.

You can simplify the orchestration pipeline by consuming change feeds.

**Using the Tables folder if your data marts must be updated daily or several times per day**

Change feeds are a powerful feature, but the process of constructing and maintaining a near-real-time data pipeline is complex. Although modern data transformation tools and ready-made templates can help simplify this process, you might still have to invest in building and running your pipeline.

If your users expect to data marts to be updated daily or several times per day, triggering a full refresh might be an economical alternative, especially if the volume of data is low or moderate.

**Changing feeds to audit and verify master data updates**

The Change feed folder is an exact replica of the CDC change logs that are maintained by the Finance and Operations database. Changes that are made to master data in Finance and Operations apps are reflected in CDC. Therefore, by extension, they are also reflected in change feed folders in your data lake.

You can use reports that are built over the Change feed folder to audit and verify master data changes in the system.

**Periodically purging the Change feed folder**

The Change feed folder isn't deleted by the Export to Data Lake process unless you reinitialize the data to recover from an error.

Because tables continue to add changes while they are in Running status, change feed folders will continue to
grow in the data lake. (However, note that the cost of retaining data in the data lake is a fraction of the cost of a SQL database. Therefore, the cost of growing data might not be a major concern.)

If you want to reduce the amount of data that is stored in your data lake, you can periodically delete the change log from the data lake. For example, you can run a job that deletes change log files that haven't been modified for 90 days or 180 days.

Periodic deletion of the change log has no impact on data in the Tables folder. However, if you run consistency checks, as described earlier in this topic, you might want to keep the change log longer to facilitate those checks.
Database movement operations are a suite of self-service actions that can be used as part of Data Application Lifecycle Management (also referred to as DataALM). These actions provide structured processes for common implementation scenarios such as golden configuration promotion, debugging/diagnostics, destructive testing, and general refresh for training purposes.

In this topic, you will learn how to use database movement operations to perform refresh, export, import, and various flavors of point-in-time restore.

### Database movement scenarios and quick start guides

The following table shows the various scenarios that are supported and a link to a quick start guide for each scenario.

<table>
<thead>
<tr>
<th>SOURCE ENVIRONMENT</th>
<th>TARGET ENVIRONMENT</th>
<th>QUICK START GUIDE</th>
<th>AVAILABLE VIA API</th>
<th>TUTORIALS</th>
</tr>
</thead>
</table>
| Production         | Sandbox            | Refresh database  | Create refresh    | Refresh for training purposes  
                    |                    |                   |                   | Debug a copy of the production database |
| Sandbox            | Production         | Refresh database  | Not supported     | Golden configuration promotion |
| Sandbox            | Sandbox            | Refresh database  | Create refresh    | Refresh for training purposes |
| Sandbox            | DevTest            | Export a database | Create export     | Export a copy of the standard user acceptance testing (UAT) database |
| DevTest            | Sandbox            | Import a database | Not supported     | Golden configuration promotion |
| Production         | DevTest            | Not directly supported | Not supported | Recommend Export a copy of the standard user acceptance testing (UAT) database |
| Sandbox point-in-time | Sandbox | Point-in-time restore (PITR) | Not supported | Destructive testing |
| Production point-in-time | Sandbox | Point-in-time restore of the production database to a sandbox environment | Not supported | Destructive testing |
Database Movement API

The Database Movement application programming interface (API) lets you integrate several of the previously mentioned database movement operations into your overall ALM process. In addition, by using the API together with your preferred scheduling engine, you can build recurrence into the process, so that it runs daily or on demand.

For more information about the Database Movement API, see the following topics:

- Overview
- Versioning and support
- Authentication
- Throttling
- Reference
The Database movement toolkit is a ZIP file hosted in Microsoft Dynamics Lifecycle Services (LCS). This toolkit is available for download. It contains a series of scripts that enhance the customer experience for movement of data between developer environments and sandbox environments. This topic explains the components of the toolkit, how to download it, and any recent changes.

Download the latest version

The toolkit is available in the LCS Shared Asset Library under the Models section. Search for the entry titled Database Movement Toolkit and then download it to your Tier1 DevTest environment.

Toolkit components

The toolkit contains the following primary components:

- **Sqlpackage.exe** - This tool is used to perform extract and publish actions against Microsoft Azure SQL databases, as well as SQL Server databases hosted on Tier1 DevTest environments.
- **PowerShell 7.0** - This is a self-contained version of PowerShell that includes capabilities for parallelism, which increases the performance of transferring data between environments.
- **PowerShell script** - There are several scripts included to provide an enhanced and more automated experience for scenarios such as AX 2012 data upgrade.

Supported scenarios

The toolkit currently supports the following scenarios. More will be added over time.

- **Upgrade from AX 2012** - Data upgrade in sandbox environments

Versions

### Version 5

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DETAILS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema publish</td>
<td>Added CleanupSource and CleanupTarget scripts.</td>
</tr>
<tr>
<td>Data transfer</td>
<td>Added timer to capture transfer time.</td>
</tr>
</tbody>
</table>

### Version 4

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DETAILS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema publish</td>
<td>Minor improvements.</td>
</tr>
</tbody>
</table>

### Version 3
<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DETAILS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema publish</td>
<td>Fixed the scripts to output errors in PublishDiag.log.</td>
</tr>
</tbody>
</table>

**Version 2**

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DETAILS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema publish</td>
<td>Fixed the scripts to use current directory reference to Sqlpackage directory.</td>
</tr>
<tr>
<td>Schema publish</td>
<td>Fixed the scripts to delete local users from the source database, and tidy up environment-specific tables.</td>
</tr>
<tr>
<td>Data transfer</td>
<td>Fixed the scripts to install missing PowerShell modules.</td>
</tr>
</tbody>
</table>

**Version 1**

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DETAILS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>Toolkit uploaded to LCS Shared Asset Library.</td>
</tr>
</tbody>
</table>
You can use Microsoft Dynamics Lifecycle Services (LCS) to perform a refresh of the database to a sandbox user acceptance testing (UAT) environment. A database refresh lets you copy the transactional and financial reporting databases of your production environment into the target, sandbox UAT environment. If you have another sandbox environment, you can also copy the databases from that environment to your target, sandbox UAT environment.

### IMPORTANT

Copying production data during business hours or peak hours could have an impact on the production system. It's highly recommended to do the refresh database operation during off-peak hours and limit only one refresh operation at a time.

Copying production data to your sandbox environment for the purpose of production reporting is not supported.

### Self-service database refresh

With the goal of providing Data Application Lifecycle Management (also referred to as DataALM) capabilities to our customers without relying on human or manual processes, the Lifecycle Services team has introduced an automated Refresh database action. This process is outlined below:

1. Visit your target sandbox on the Environment Details page, and click the Maintain > Move database menu option.
2. Select the Refresh database option and choose your source environment.
3. Note the warnings and review the list of data elements that are not copied from the source environment.
4. The refresh operation will begin immediately.

**Refresh operation failed**

In case of failure, the option to perform a rollback is available. By clicking the Rollback option after the operation has initially failed, your target sandbox environment will be restored to the state it was before the refresh began. This is made possible by the Azure SQL point-in-time restore capability to restore the database. This is often required if a customization, that is present in the target sandbox, cannot complete a database synchronization with the newly refreshed data.

To determine the root cause of the failure, use the available buttons to download the runbook logs before you start the rollback operation.

**Data elements that aren't copied during refresh**

The information in this section lists certain elements of the database that are not copied over to the target environment during a database refresh operation.

When refreshing a production environment to a sandbox environment or a sandbox to another sandbox environment

- Email addresses in the LogisticsElectronicAddress table.
- SMTP Relay server in the SysEmailParameters table.
- Print Management settings in the PrintMgmtSettings and PrintMgmtDocInstance tables.
- All users except the admin will be set to Disabled status.

When refreshing from sandbox environment to production environment

This is also referred to as Golden configuration promotion.
These elements are removed for all database refresh operations

- All files stored in Azure blob storage. This includes document attachments (from the DocuValue and DocuDeletedValue tables) and custom Microsoft Office templates (from the DocuTemplate table).
- All batch jobs are set to Withhold status.
- All users will have their partition value reset to the “initial” partition record ID.
- All Microsoft-encrypted fields will be cleared, because they can’t be decrypted on a different database server. An example is the Password field in the SysEmailSMTPPassword table.
- Maintenance mode settings will be disabled even if it was enabled in source.
- Dual-write configuration. To setup a new link on the target environment after this operation is successful, see Dual-write environment linking.

Some of these elements aren’t copied because they are environment-specific. Examples include BatchServerConfig and SysCorpNetPrinters records. Other elements aren’t copied because of the volume of support tickets. For example, duplicate emails might be sent because Simple Mail Transfer Protocol (SMTP) is still enabled in the UAT environment, invalid integration messages might be sent because batch jobs are still enabled, and users might be enabled before admins can perform post-refresh cleanup activities.

Environment administrator

The System Administrator account in the target environment (UserId of ‘Admin’) is reset to the value found in the web.config file on the target. This should be the same value as that of the Administrator from Lifecycle Services. To preview which account this will be, visit your target sandbox Environment Details page in LCS. The value of the Environment Administrator field that was selected when the environment was first deployed is updated to be the System Administrator in the transactional database. This also means that the tenant of the environment will be that of the Environment Administrator.

If you have used the Admin User Provisioning Tool on your environment to change the web.config file to a different value, it may not match what is in Lifecycle Services. If you require a different account to be used, you will need to deallocate and delete the target sandbox, and redeploy selecting another account. After this, you can perform another refresh database action to restore the data.

An environment can’t be refreshed from one tenant to another. This restriction applies even to .onmicrosoft.com tenants. You should make sure that the admin accounts in the source and target environments are from the same tenant domain.

Conditions of a database refresh

Here is the list of requirements and conditions of operation for a database refresh:

- A refresh performs a delete operation on the original target database.
- The target environment will be available until the database copy has reached the target server. After that point, the environment will be offline until the refresh process is completed.
- The refresh will affect only the application and Financial Reporting databases.
- No file stored in Azure blob storage is copied from one environment to another. This includes document attachments and custom Microsoft Office templates. These documents won’t be changed and will remain in their current state.
- All users except the Admin user and other internal service user accounts will be unavailable. This process allows the Admin user to delete or obfuscate data before allowing other users back into the system.
- The Admin user must make required configuration changes, such as reconnecting integration endpoints to specific services or URLs.
- All data management framework with recurring import and export jobs must be fully processed and stopped
in the target system prior to initiating the restore. In addition, we recommend that you select the database from the source after all recurring import and export jobs have been fully processed. This will ensure there are no orphaned files in Azure storage from either system. This is important because orphaned files cannot be processed after the database is restored in the target environment. After the restore, the integration jobs can be resumed.

- Any user with a role of Project owner or Environment manager in LCS will have access to the SQL and machine credentials for all non-production environments.
- The databases must be hosted in the same Azure geographic region, unless the databases are Spartan-managed. Databases are Spartan-managed when you see ‘spartan’ as part of the fully qualified SQL server address.
- The allocated database capacity of the source environment must be less than the maximum database capacity of the target environment.

Steps to complete after a database refresh for environments that use Commerce functionality

**IMPORTANT**

Some environment-specific records are not included in automated database movement operations and require additional steps. These include the following:

- Commerce self-service installer references
- Commerce Scale Unit channel database configuration records

If you copy a database between environments, Commerce capabilities in the destination environment will not be fully functional until you perform the following additional steps.

**Initialize Commerce Scale Units**

If you are moving a database to a sandbox UAT or production environment, you must Initialize Commerce Scale Unit after the database movement operation is complete. The Commerce Scale Unit’s association from the source environment will not copy over to the destination environment.

**Synchronize Commerce self-service installers**

To be able to access Commerce self-service installers in HQ, you must Synchronize self-service installers after the database movement operation is complete.

**IMPORTANT**

The Environment re-provisioning step has now been fully automated as part of database movement operations, and no longer needs to be run manually. The Environment re-provisioning tool is still available in the Asset Library and may be used in certain situations to mitigate error conditions.

To run the Environment re-provisioning tool on the destination environment, run the following steps:

1. In your project’s Asset Library, in the Software deployable packages section, select Import.
2. From the list of shared assets, select the Environment Reprovisioning Tool.
3. On the Environment details page for your destination environment, select Maintain > Apply updates.
4. Select the Environment Reprovisioning tool that you uploaded earlier, and then select Apply to apply the package.
5. Monitor the progress of the package deployment.

For more information about how to apply a deployable package, see [Create deployable packages of models](#). For
Re-activate POS devices

If you use point of sale (POS) devices, after you import a database you must activate the POS devices again. Previously activated devices in the destination environment will no longer function. For more information, see Point of sale device activation.

Known issues

The Restore operation fails if the sandbox customizations are incompatible with production data

Even if a customization is successfully added to the sandbox environment (that is, the customer’s AOT deployable package is successfully installed via LCS), it might not succeed for production data. For example, a customer adds a unique index on **Vendor Name** to the VendTable table. This customization can be successfully installed if there are no duplicate vendor names in the sandbox environment. However, when the production database is brought in as part of the Restore operation, installation might fail if there are duplicates in the dataset that is inbound to the sandbox environment. Duplicates in this dataset aren't supported. Therefore, you must remove the customization before you can have a successful Restore operation.

Refresh is denied for environments that run Platform update 20 or earlier

The database refresh process can't currently be completed if the environment is running Platform update 20 or earlier. For more information, see the list of currently supported platform updates.

Incompatible version of Financial Reporting between source and target environments

The database refresh process (self-service or via a service request) can't be completed successfully if the version of Financial Reporting in the target environment is earlier than the version in the source environment. To resolve this issue, update both environments so that they have the latest version of Financial Reporting.

To determine the version you have installed in your source and target environments, visit the View detailed version information link on the Environment Details page.

Search for **MRApplicationService** and ensure that the target environment is greater than or equal to the source environment.

For customers that are using version 8.1 or later:

1. Go to the Update tiles for your UAT environment. Save the updates to your Project asset library.
2. Apply this package to your UAT environment.
3. Verify that the error has been resolved.

For customers that are using version 8.0 or earlier:

1. Review the Environment history of your source environment. Specifically, look for any "Platform and application binary package" that might have been deployed to the source environment and not the target environment.
2. Apply this binary package to your target environment.
3. Verify that the error has been resolved.
Incompatible application versions between source and target environments

The database refresh process (self-service or via service request) cannot be completed if the Application release of your source and target environment are not the same. This is because the data upgrade process is not executed by database movement operations such as refresh, and data loss can occur.

If upgrading your sandbox UAT environment to a newer Application version (for example, 7.3 to 8.1), be sure to perform the database refresh action prior to starting the upgrade. After your sandbox is upgraded to the newer version, you cannot restore an older production environment database in to the sandbox UAT environment.

Conversely, if your production environment is newer than your target sandbox, you will need to either upgrade the target sandbox prior to the refresh or simply deallocate, delete, and redeploy prior to performing the refresh.
You can use Microsoft Dynamics Lifecycle Services (LCS) to export a database from a sandbox user acceptance testing (UAT) environment to the Asset library.

**Self-service export database**

From your sandbox **Environment Details** page, click the **Maintain** menu, and then select **Move database**.

A slider pane will open on the page where you can use the **Export database** action.
The environment will be unavailable for other servicing operations, such as Sandbox refresh or package deployment during this time. The source environment will be usable from a Dynamics user perspective.

After the export operation completes successfully, sign out of the servicing operation on your Environment details page. You can then see the asset in your Asset Library in the Database backups section.
The .bacpac files are stored here and can be manually downloaded to your Tier 1 developer environments for import. In the future, Microsoft will provide APIs to trigger the export action, as well as list the available backup files in your asset library. This includes the secured URL for automatically downloading a backup asset file or copying it directly to your secure blob storage using Microsoft Azure Storage SDKs.

**Maximum limit 50 GB on exported bacpacs**

To maintain the system that performs database export from LCS, a limit on the maximum bacpac size is being imposed. This limit is set at 50 GB for each bacpac exported. The reasons for this limit include:

- A centralized system is performing the exports for multiple customers in the same geographic region, and this system has constraints on disk space.
- Azure SQL neatly compresses the data in the bacpac format and in many cases, where customers exceed 50 GB, customizations or binary data drastically oversize the backup file.

If you need to reduce the size of the database, follow the cleanup routines.

If the above cleanup routines did not help to bring the bacpac file to under 50 GB in size, try running the following SQL script against your sandbox database to identify the top 15 tables by size in megabytes. Any tables for data entity staging (they will have "staging" at the end of the table name) can be truncated. Any tables that store binary or blob data (JSON/XML/binary) should either be truncated or the contents of that field should be deleted to free up space. Binary data cannot be compressed, so storing large volumes of data in the database itself will cause you to quickly reach the 50 GB limit.

```sql
USE [YourDBName] -- replace your dbname
GO
SELECT top 15
  s.Name AS SchemaName,
  t.Name AS TableName,
  p.Rows AS RowCounts,
  CAST(ROUND((SUM(a.used_pages) / 128.00), 2) AS NUMERIC(36, 2)) AS Used_MB,
  CAST(ROUND((SUM(a.total_pages) - SUM(a.used_pages)) / 128.00, 2) AS NUMERIC(36, 2)) AS Unused_MB,
  CAST(ROUND((SUM(a.total_pages) / 128.00), 2) AS NUMERIC(36, 2)) AS Total_MB
FROM sys.tables t
  INNER JOIN sys.indexes i ON t.OBJECT_ID = i.object_id
  INNER JOIN sys.partitions p ON i.object_id = p.OBJECT_ID AND i.index_id = p.index_id
  INNER JOIN sys.allocation_units a ON p.partition_id = a.container_id
  INNER JOIN sys.schemas s ON t.schema_id = s.schema_id
GROUP BY t.Name, s.Name, p.Rows
ORDER BY Total_MB DESC
GO
```

**Export operation failure**

Most often, export operations fail because the process in LCS times out while it's waiting for a response from Microsoft Azure SQL Database. You can use the **Resume** button to reconnect LCS to the ongoing export process and see it through to completion. If more than 24 hours have passed since you began the export, the pending asset in the LCS Project asset library will be expired. In this case, you must roll back the export operation and restart it.

To cancel an export operation that has failed, you can use the **Rollback** button.

**Data elements that aren’t exported**

When you export a database backup from an environment, some elements of the database aren’t exported in the backup file. Here are some examples:

- Email addresses in the **LogisticsElectronicAddress** table.
- Batch job history in the **BatchJobHistory**, **BatchHistory**, and **BatchConstraintsHistory** tables.
- SMTP Relay server in the **SysEmailParameters** table.
Print Management settings in the `PrintMgmtSettings` and `PrintMgmtDocInstance` tables.


Document attachments in the `DocuValue` table. These attachments include any Microsoft Office templates that were overwritten in the source environment.

All users except the admin will be set to **Disabled** status.

All batch jobs are set to **Withhold** status.

All users will have their partition value reset to the “initial” partition record ID.

All Microsoft-encrypted fields will be cleared, because they can’t be decrypted on a different database server. An example is the `Password` field in the `SysEmailSMTPPassword` table.

Maintenance mode settings will be disabled even if it was enabled in source.

Dual-write configuration. To setup a new link on the target environment after this operation is successful, see [Dual-write environment linking](#).

---

### Known issues

#### Export ran for some time and then reached a “Preparation failed” state

The export process can time out on Azure SQL Database when large databases are involved. In some cases, the export process can be recovered by using the **Resume** action from LCS. The Lifecycle Services team is working to identify known error codes, so these can be added to the logs for the export database operation to help guide users toward a resolution. These known error codes will be added in a future release of LCS.

#### Export doesn’t show any progress in LCS

The export process differs from other database movement operations, and the general package deployment doesn’t use a runbook. Therefore, the progress indicator in LCS doesn’t show any output, even though it typically shows output in other scenarios. The LCS team is working to identify known error codes so that they can be added to the logs for the export database operation to help guide users toward a resolution. These known error codes will be added in a future release of LCS.
You can use Microsoft Dynamics Lifecycle Services (LCS) to import a golden configuration database into a sandbox user acceptance testing (UAT) environment.

**Prerequisites**

Database import isn't applicable to LCS projects that are configured for a Dynamics AX 2012 upgrade. Therefore, import will be blocked if the Legacy system field at Project Onboarding > Project overview is set to AX2012 Upgrade.

**Self-service import database**

To import a database that is prepared from a developer environment to a standard user acceptance test (UAT), or a database previously exported from a UAT environment, follow the steps outlined below:

1. Go to your target sandbox Environment Details page, and select the Maintain > Move database menu option.
2. Select Import database and choose your source database backup (.bacpac format) file from the Asset Library.
3. Note the warnings. Review the list of data elements that are cleaned up from the backup file.
4. The import operation will begin immediately.

**NOTE**

All users except the Admin user and other internal service user accounts will be unavailable after import. Therefore, the Admin user can delete or obfuscate data before other users are allowed back into the system.

To import a database to a developer environment after you've downloaded a database backup (.bacpac) file, you can begin the manual import operation on your Tier 1 environment. When you import the database, we recommend that you follow these guidelines:

- Keep a copy of the existing AxDB database, so that you can revert to it later if needed.
- Import the new database under a new name, such as AxDB_fromProd.

To ensure the best performance, copy the *.bacpac file to the local computer that you're importing from. Download sqlpackage .NET Core for Windows from Get sqlpackage .NET Core for Windows. Open a Command Prompt window, and run the following commands from the sqlpackage .NET Core folder.

```
SqlPackage.exe /a:import /sf:D:\Exportedbacpac\my.bacpac /tsn:localhost /tdn:<target database name> /p:CommandTimeout=1200
```

Here is an explanation of the parameters:

- **tsn (target server name)** – The name of the Microsoft SQL Server instance to import into.
- **tdn (target database name)** – The name of the database to import into. The database should not already exist.
- **sf (source file)** – The path and name of the file to import from.
NOTE
During import, the user name and password aren't required. By default, SQL Server uses Microsoft Windows authentication for the user who is currently signed in.

For information about how to complete the manual import operations into a Tier 1 environment, see Import the database.

Import operation failure
If the import operation isn't successful, you can do a rollback. If you select the Rollback option after the initial failure of the operation, your target sandbox environment is restored to the state that it was in before the import began. The rollback operation is made available by the Microsoft Azure SQL Database point-in-time restore capability for restoring the database. Rollback is often required if a customization that is present in the target sandbox can't complete a database synchronization with the newly imported data.

Data elements that require attention after import
Specific activities must be completed when you import a database backup into a sandbox UAT environment. Here are some examples:

- Make sure that the source database contains only a single record in the Partitions table.
- Make sure that email capabilities are correctly reconfigured or turned off, according to your requirements.
- Make sure that integration settings are turned on or off, according to your requirements.
- Make sure that Application Object Server (AOS) servers are added back to required batch groups.
- Make sure that system Help and task guides are reconnected.
- Make sure that batch jobs are set to a status of Waiting.
- Make sure that users are re-enabled.
- Make sure that dual-write is relinked if required.
- Make sure that dual-write is relinked if required. To setup a new link on the target environment after this operation is successful, see Dual-write environment linking.

Environment admin
The system admin account in the target environment (Admin user ID) is reset to the value that is found in the web.config file in that environment. This account should be the same as the admin account from LCS. To preview which account this account will be, visit the Environment details page for your target sandbox in LCS. The value that was selected in the Environment Administrator field when the environment was first deployed is updated to the system admin in the transactional database. Therefore, the tenant of the environment will be the tenant of the environment admin.

If you've used the Admin User Provisioning Tool on your environment to change the web.config file, the value might not match the value in LCS. If you require that a different account be used, you must deallocate and delete the target sandbox, redeploy, and select another account. You can then do another database refresh action to restore the data.

Steps to complete after a database import for environments that use Commerce functionality
IMPORTANT
Some environment-specific records are not included in automated database movement operations and require additional steps. These include the following:

- Commerce self-service installer references
- Commerce Scale Unit channel database configuration records

If you copy a database between environments, Commerce capabilities in the destination environment will not be fully functional until you perform the following additional steps.

**Initialize Commerce Scale Units**
If you are moving a database to a sandbox UAT or production environment, you must Initialize Commerce Scale Unit after the database movement operation is complete. The Commerce Scale Unit's association from the source environment will not copy over to the destination environment.

**Synchronize Commerce self-service installers**
To be able to access Commerce self-service installers in HQ, you must Synchronize self-service installers after the database movement operation is complete.

IMPORTANT
The Environment re-provisioning step has now been fully automated as part of database movement operations, and no longer needs to be run manually. The Environment re-provisioning tool is still available in the Asset Library and may be used in certain situations to mitigate error conditions.

To run the Environment re-provisioning tool on the destination environment, run the following steps:

1. In your project’s Asset Library, in the Software deployable packages section, select Import.
2. From the list of shared assets, select the Environment Reprovisioning Tool.
3. On the Environment details page for your destination environment, select Maintain > Apply updates.
4. Select the Environment Reprovisioning tool that you uploaded earlier, and then select Apply to apply the package.
5. Monitor the progress of the package deployment.

For more information about how to apply a deployable package, see Create deployable packages of models. For more information about how to manually apply a deployable package, see Install deployable packages from the command line.

**Re-activate POS devices**
If you use point of sale (POS) devices, after you import a database you must activate the POS devices again. Previously activated devices in the destination environment will no longer function. For more information, see Point of sale device activation.
You can use Microsoft Dynamics Lifecycle Services (LCS) to perform the point-in-time restore (PITR) for a sandbox user acceptance testing (UAT) environment. Microsoft maintains automated backups of the business and financial reporting databases for 28 days for production environments and 7 days for sandbox environments.

Self-service point-in-time restore

To restore the database of a standard user acceptance test (UAT) environment to a previous point-in-time, follow the steps outlined below:

1. Go to your target sandbox Environment Details page, and select the Maintain > Move database menu option.
2. Select the Point-in-time restore option and choose a point-in-time.
3. Note the warnings. Review the list of data elements that are not copied over from the previous point-in-time.
4. The restore operation will begin immediately.

Restore operation failed

In the event of failure, the option to do a rollback is available. If you select the rollback option after the operation originally fails, your target sandbox environment is restored to the state that it was in before the restore began. A rollback is often required if a customization that is present in the target sandbox environment can't complete a database synchronization with the newly restored data.

To determine the root cause of the failure, download the runbook logs using the available buttons before starting the rollback operation.

Data elements that need attention after restore

When you restore a database from a previous point in time, keep in mind that the database is provided “as was.” For example, batch jobs and other data elements in the system can be in an in-progress state. These elements will require manual review.

NOTE

Restore does affect the tables that store references to files stored in Azure blob storage (like document attachments and custom Microsoft Office templates). However, as Azure blob storage itself is not affected by this process, any files added after the restore point will continue to exist in Azure blob storage but will not be reflected in the database.

Environment administrator

The system administrator account in the target environment (that is, the account that has a UserId value of Admin) is reset to the value that is found in the web.config file in the target environment. This value should match the value of the administrator account in LCS. To preview which account will be used, go to the Environment Details page for your target sandbox environment in LCS. The value that was selected in the Environment Administrator field when the environment was first deployed is updated to the system administrator in the transactional database. The tenant of the environment will be the tenant of the environment administrator.

If you've used the Admin User Provisioning Tool in your environment to change the value in the web.config file, the value might not match the value in LCS. If you require a different account, you must deallocate and delete
the target sandbox environment, and then redeploy it by selecting another account. You can then do another refresh database action to restore the data.

Steps to complete after a database restore for environments that use Commerce functionality

**IMPORTANT**

Some environment-specific records are not included in automated database movement operations and require additional steps. These include the following:

- Commerce self-service installer references
- Commerce Scale Unit channel database configuration records

If you copy a database between environments, Commerce capabilities in the destination environment will not be fully functional until you perform the following additional steps.

**Initialize Commerce Scale Units**

If you are moving a database to a sandbox UAT or production environment, you must Initialize Commerce Scale Unit after the database movement operation is complete. The Commerce Scale Unit's association from the source environment will not copy over to the destination environment.

**Synchronize Commerce self-service installers**

To be able to access Commerce self-service installers in HQ, you must Synchronize self-service installers after the database movement operation is complete.

**IMPORTANT**

The Environment re-provisioning step has now been fully automated as part of database movement operations, and no longer needs to be run manually. The Environment re-provisioning tool is still available in the Asset Library and may be used in certain situations to mitigate error conditions.

To run the Environment re-provisioning tool on the destination environment, run the following steps:

1. In your project's Asset Library, in the Software deployable packages section, select Import.
2. From the list of shared assets, select the Environment Reprovisioning Tool.
3. On the Environment details page for your destination environment, select Maintain > Apply updates.
4. Select the Environment Reprovisioning tool that you uploaded earlier, and then select Apply to apply the package.
5. Monitor the progress of the package deployment.

For more information about how to apply a deployable package, see Create deployable packages of models. For more information about how to manually apply a deployable package, see Install deployable packages from the command line.

**Re-activate POS devices**

If you use point of sale (POS) devices, after you import a database you must activate the POS devices again. Previously activated devices in the destination environment will no longer function. For more information, see Point of sale device activation.

**Known issues**

**Breaking the chain of available restore points**
Several frequently used actions create a new database that won't have the same restore point history as the previously used database. These actions include point-in-time restores, database refreshes, database imports, and point-in-time restores from the production environment to the sandbox environment. In addition, if a software deployable package that you apply to your environment fails during update of the database, and you use the rollback functionality in LCS, the rollback functionality does a point-in-time restore of the database, and that restore also creates a new database.

Although the new database doesn't have any restore point history, it does begin to accrue new restore points from that time onward. After you perform any of the previously mentioned actions, you can't perform it again by using the same restore date and time.

**Restore is denied for environments that run Platform update 20 or earlier**

The restore database process cannot be completed if the environment is running Platform update 20 or earlier. For more information, see the list of currently supported Platform updates in the Software lifecycle policy and cloud releases.
You can use Microsoft Dynamics Lifecycle Services (LCS) to do a point-in-time restore (PITR) of the production database to a user acceptance testing (UAT) sandbox environment. Microsoft maintains automated backups of the business and financial reporting databases for 28 days for Production environments and 7 days for Sandbox environments.

**IMPORTANT**

Microsoft doesn't support copying production data to a sandbox environment for the purpose of production reporting.

### Self-service point-in-time restore Production to Sandbox

To provide customers with data application lifecycle management (DataALM) capabilities that don’t rely on human or manual processes, the Lifecycle Services team has introduced an automated refresh database action. Here is an overview of the process for doing a self-service database refresh.

1. Go to the **Environment Details** page for your target Sandbox, and select **Maintain > Move database** button.
2. Select the **Point-in-time restore Prod to Sandbox** option, and then select the desired point in time.
3. Make a note of the warnings, and review the list of data elements that aren’t copied from the source environment’s previous point in time.
4. The restore operation begins immediately.

**IMPORTANT**

Self-service point in time restore (PITR) is not supported between environments that are on different regions. For more details, refer to the “Known issues” section later in this topic.

### Restore operation failure

If the restore operation isn’t successful, you can do a rollback. If you select the **Rollback** option after the operation originally fails, your target sandbox environment is restored to the state that it was in before the refresh began. Rollbacks are made available by the PITR capability of Azure SQL Database. They are typically required if a customization that is present in the target sandbox environment can’t complete a database synchronization with the newly refreshed data.

To determine the root cause of the failure, use the available buttons to download the runbook logs before you start the rollback operation.

### Data elements that aren't copied during restore copy

When you refresh a production environment to a sandbox environment, or a sandbox environment to another sandbox environment, some elements of the database aren’t copied over to the target environment. Here are some examples:

- Email addresses in the LogisticsElectronicAddress table.
- Batch job history in the BatchJobHistory, BatchHistory, and BatchConstraintHistory tables.
- Simple Mail Transfer Protocol (SMTP) Relay server in the SysEmailParameters table.
- Print Management settings in the PrintMgmtSettings and PrintMgmtDocInstance tables.
- All files stored in Azure blob storage. This includes document attachments (from the DocuValue and DocuDeletedValue tables) and custom Microsoft Office templates (from the DocuTemplate table).
- All users except the admin will be set to **Disabled** status.
- All batch jobs will be set to **Withhold** status.
- All users will have their partition value reset to the "initial" partition record ID.
- All Microsoft-encrypted fields will be cleared, because they can't be decrypted on a different database server. An example is the **Password** field in the SysEmailSMTPPassword table.
- Dual-write configuration. To setup a new link on the target environment after this operation is successful, see **Dual-write environment linking**.

Some of these elements aren't copied because they are environment-specific. Examples include BatchServerConfig and SysCorpNetPrinters records. Other elements aren't copied because of the volume of support tickets. For example, duplicate emails might be sent because SMTP is still turned on in the UAT environment, invalid integration messages might be sent because batch jobs are still enabled, and users might be enabled before admins can perform post-refresh cleanup activities.

**Environment administrator**

The system administrator account in the target environment (that is, the account that has a **UserId** value of **Admin**) is reset to the value that is found in the web.config file in the target environment. This value should match the value of the administrator account in LCS. To preview this account, go to the **Environment Details** page for your target sandbox environment in LCS. The value that was selected in the **Environment Administrator** field when the environment was first deployed is updated so that it matches the system administrator account in the transactional database. Therefore, the tenant of the environment will also be the tenant of the environment administrator.

If you've used the Admin User Provisioning Tool on your environment to change the value in the web.config file, that value might not match the value in LCS. If you require that a different account be used, you must deallocate and delete the target sandbox environment, and then redeploy it and select another account. You can then perform another refresh database action to restore the data.

An environment can't be refreshed from one tenant to another. This restriction applies even to .onmicrosoft.com tenants. You should make sure that the admin accounts in the source and target environments are from the same tenant domain.

**Conditions for doing a PITR copy of a production environment to a sandbox environment**

Here is the list of requirements and conditions of operation for a database refresh:

- A refresh performs a delete operation on the original target database.
- The target environment will be available until the database copy has reached the target server. After that point, the environment will be offline until the refresh process is completed.
- The refresh will affect only the application and Financial Reporting databases.
- **No file stored in Azure blob storage is copied** from one environment to another. This includes document attachments and custom Microsoft Office templates. These documents won't be changed and will remain in their current state.
- All users except the Admin user and other internal service user accounts will be unavailable. Therefore, the Admin user can delete or obfuscate data before other users are allowed back into the system.
- The Admin user must make required configuration changes, such as reconnecting integration endpoints to specific services or URLs.
- All data management integration jobs that have recurring import and export enabled must be fully processed and stopped in the target system before the restore is started. In addition, we recommend that you select the database from the source after all recurring import and export jobs have been fully processed. In this way, you ensure that there are no orphaned files from either system in Azure storage. This step is important because orphaned files can’t be processed after the database is restored in the target environment. After the restore, the integration jobs can be resumed.

- Business events end points must be deleted and reconfigured in the environment where the database is restored to ensure the same end points are not used. This will also require the business events to be deactivated and re-activated to the new end points that were configured in the environment.

- Any user who has the Project owner or Environment manager role in LCS will have access to the SQL and machine credentials for all non-production environments.

- The databases must be hosted in the same Azure geographic region, unless the databases are Spartan-managed. Databases are Spartan-managed when you see ‘spartan’ as part of the fully qualified SQL server address.

- The allocated database capacity of the source environment must be less than the maximum database capacity of the target environment.

### Steps to complete after a restore is done for environments that use Commerce functionality

**IMPORTANT**

Some environment-specific records are not included in automated database movement operations and require additional steps. These include the following:

- Commerce self-service installer references
- Commerce Scale Unit channel database configuration records

If you copy a database between environments, Commerce capabilities in the destination environment will not be fully functional until you perform the following additional steps.

**Initialize Commerce Scale Units**

If you are moving a database to a sandbox UAT or production environment, you must Initialize Commerce Scale Unit after the database movement operation is complete. The Commerce Scale Unit's association from the source environment will not copy over to the destination environment.

**Synchronize Commerce self-service installers**

To be able to access Commerce self-service installers in HQ, you must Synchronize self-service installers after the database movement operation is complete.

**IMPORTANT**

The Environment re-provisioning step has now been fully automated as part of database movement operations, and no longer needs to be run manually. The Environment re-provisioning tool is still available in the Asset Library and may be used in certain situations to mitigate error conditions.

To run the Environment re-provisioning tool on the destination environment, run the following steps:

1. In your project's Asset Library, in the Software deployable packages section, select Import.
2. From the list of shared assets, select the Environment Reprovisioning Tool.
3. On the Environment details page for your destination environment, select Maintain > Apply updates.
4. Select the Environment Reprovisioning tool that you uploaded earlier, and then select Apply to apply the
Re-activate POS devices

If you use point of sale (POS) devices, after you import a database you must activate the POS devices again. Previously activated devices in the destination environment will no longer function. For more information, see Point of sale device activation.

Known issues

The Restore operation fails if the sandbox customizations are incompatible with production data

Even if a customization is successfully added to the sandbox environment (that is, the customer’s AOT deployable package is successfully installed via LCS), it might not succeed for production data. For example, a customer adds a unique index on Vendor Name to the VendTable table. This customization can be successfully installed if there are no duplicate vendor names in the sandbox environment. However, when the production database is brought in as part of the Restore operation, installation might fail if there are duplicates in the dataset that is inbound to the sandbox environment. Duplicates in this dataset aren’t supported. Therefore, you must remove the customization before you can have a successful Restore operation.

The Restore operation is denied for environments that run Platform update 20 or earlier

The database refresh process can’t currently be completed if the environment is running Platform update 20 or earlier. For more information, see the list of currently supported platform updates.

The source and target environments have incompatible versions of Financial Reporting

The database refresh process (self-service or via a service request) can’t be successfully completed if the version of Financial Reporting in your target environment is earlier than the version in your source environment. To resolve this issue, update both environments so that they have the latest version of Financial Reporting.

To determine the version that is installed in your source and target environments, select View detailed version information on the Environment Details page. Then search for MRApplicationService, and make sure that the version in the target environment is later than or the same as the version in the source environment.

Customers that are using version 8.1 or later should follow these steps.

1. Go to the Update tiles for your UAT environment. Save the updates to your Project asset library.
2. Apply the package to your UAT environment.
3. Verify that the error has been resolved.

Customers that are using version 8.0 or earlier should follow these steps.

1. Review the environment history of your source environment. Specifically, look for any “Platform and application binary package” package that has been deployed to the source environment but not to the target environment. Monitor the progress of the package deployment.

For more information about how to apply a deployable package, see Create deployable packages of models. For more information about how to manually apply a deployable package, see Install deployable packages from the command line.
2. Apply the binary package to your target environment.
3. Verify that the error has been resolved.

The source and target environments have incompatible application versions
The database refresh process (self-service or via service request) can't be completed if the application release of your source environment and the application release of your target environment aren't the same. Because the data upgrade process isn't run by using database movement operations such as refresh, data loss can occur.

If you're upgrading your sandbox UAT environment to a newer application version (for example, from 7.3 to 8.1), be sure to perform the database refresh action before you start the upgrade. After your sandbox environment is upgraded to the newer version, you can't restore an older production environment database to the sandbox UAT environment.

Conversely, if your production environment is newer than your target sandbox environment, you must either upgrade the target sandbox environment before the refresh, or just deallocate, delete, and redeploy the environment before you do the refresh.

The source and target are on different infrastructure (Microsoft-managed vs. self-Service)
The PITR process is not supported between a Microsoft-managed environment as a source and a self-service environment as a destination. For example, if the production environment is Microsoft-managed and in East US, and a PITR is needed for the sandbox environment, which is self-service and in East US. PITR is not supported. The alternative is to move the production environment to self-service or opt for a regular database refresh instead.

Point in time restore between source and target that are both on self-Service, in different regions
The PITR process is not supported between self-service environments across different regions. For example, if the production environment is in East US and a PITR is needed for the sandbox environment, which is self-service and in West Europe, PITR is not supported. The alternative is to get both the environments in the same region or opt for a regular database refresh instead.
Enable just-in-time database access

This topic provides the steps necessary to enable database access using a just-in-time (JIT) fashion. This is useful if access to the database is required for various troubleshooting efforts, running unplanned queries, or data upgrade problem solving. This process is available for both self-service as well as Microsoft-managed sandbox acceptance test environments. For more information about the available environment types, see Deployment overview.

Microsoft-managed environments without RDP access

If you no longer have Remote Desktop Protocol (RDP) access to your sandbox, you can add your IP address to the allow-list in a self-service manner from Lifecycle Services (LCS). When RDP is removed from an environment, the machine credentials section of the environment details page is removed. This leaves just the database accounts section, as shown in the following screenshot.

From the environment details page for your sandbox environment, select Maintain > Enable access, and then in the dialog box, add the IP address of your source environment. This entry will expire, with the expire date shown alongside the IP address you entered. It also will be lost after the database is replaced by a database movement operation, such as database refresh or database import.

You can now use tools like SQL Server Management Studio (SSMS) to connect to the database, using the accounts from LCS and the IP address that you enabled. Note that LCS shows the server and database in the following format: serverName\databaseName. To connect in SSMS, you will need to append the domain name suffix, such as serverName.database.windows.net if you are in Azure public cloud. On the Options tab in the SSMS connection window, you will also need to explicitly enter the databaseName value in the Database field to successfully connect.

Self-service environments

The self-service environment type has never had Remote Desktop Protocol (RDP) access or static database accounts. However, it is still possible to access the database.

From the environment details page for your sandbox environment, select Maintain > Enable access, and then
in the dialog box, add the IP address of your source environment. This firewall entry will expire after 8 hours or it will be lost after the database is replaced by a database movement operation (whichever comes first). This includes operations such as database refresh or database import.

You also need to enter which type of access you require in the **Database Accounts** section. The available options include read or read-write access. Enter a short reason description and then select **Request access**.

```
DATABASE ACCOUNTS
Reason for access                  Details
Performance tuning for AX (write to AX)          Need to test data hotfix

Request Access
```

When the page is refreshed, the database account will be shown with its expiry time.

```
DATABASE ACCOUNTS
Reason for access                  Details

Request Access
```

![Database Account Example](serverName\databaseName)

You can now use tools like SQL Server Management Studio (SSMS) to connect to the database, using the accounts from LCS and the IP address that you enabled. Note that LCS shows the server and database in the following format: `servername\databaseName`. To connect in SSMS, you will need to append the domain name suffix, such as `servername.database.windows.net` if you are in Azure public cloud. On the **Options** tab in the SSMS connection window, you will also need to explicitly enter the `databaseName` value in the **Database** field to successfully connect.
Database movement operations are a suite of self-service actions that can be used as part of data application lifecycle management (DataALM). This tutorial shows how to use the refresh database operation in a training scenario.

In this tutorial, you will learn how to:

- Prepare the target environment.
- Run the refresh.
- Reconfigure the target environment.
- Enable selected users.

As an example of this scenario, a customer who has already gone live with the application wants to load a recent copy of production transactions into the user acceptance testing (UAT) environment. In this way, the customer can support training of new employees and evaluate configuration changes without affecting the live environment.

**Prerequisites**

To do a refresh database operation, your production environment must be deployed, or you must have a minimum of two standard UAT environments.

**Notify users about the pending downtime**

Before you start the bulk of the work, notify users of the target environment that the environment will be offline for a period. You can notify users either manually via Microsoft Dynamics Lifecycle Services (LCS) or programmatically by using RESTful application programming interface (API) calls.

**Manually send a broadcast message**

To notify users manually via LCS, follow these steps.

1. In LCS, open the **Environment details** page for the target environment.
2. Select **Maintain > Message online users**.
3. Select **Broadcast a new message for downtime**.
4. Select the valid from/valid to times in your local time zone.
5. Select **Post**.

**Programmatically send a broadcast message**

The following sample code can be used as shown in a console application, or it can be modified to make it work with other services that can be called on demand, such as Microsoft Azure Functions. Before this sample code will work, you must set up your application registration as described in Service endpoints overview.
[Serializable]
public class SysAddBroadcastMessageDataContract
{
    public SysAddBroadcastMessageRequest request { get; set; }
}

[Serializable]
public class SysAddBroadcastMessageRequest
{
    public DateTime FromDateTime { get; set; }
    public DateTime ToDateTime { get; set; }
}

public class Program
{
    public static AuthenticationResult getResult(AuthenticationContext ctx, string url)
    {
    }
    static void Main(string[] args)
    {
        SysAddBroadcastMessageRequest request = new SysAddBroadcastMessageRequest();
        request.FromDateTime = DateTime.UtcNow;
        request.ToDateTime = DateTime.UtcNow.AddHours(12);

        SysAddBroadcastMessageDataContract dc = new SysAddBroadcastMessageDataContract();
        dc.request = request;

        AuthenticationResult res = getResult(ctx, "https://YOUR_SANDBOX_UAT.sandbox.operations.dynamics.com");

        HttpClient restfulCli = new HttpClient();
        restfulCli.DefaultRequestHeaders.Clear();
        restfulCli.BaseAddress = new Uri("https://YOUR_SANDBOX_UAT.sandbox.operations.dynamics.com/");
        restfulCli.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

        HttpRequestMessage requestMsg = new HttpRequestMessage(HttpMethod.Post, string.Format("api/services/SysBroadcastMessageServices/SysBroadcastMessageService/AddMessage"));
        requestMsg.Content = new StringContent(JsonConvert.SerializeObject(dc));

        HttpResponseMessage responseMsg = restfulCli.SendAsync(requestMsg).Result;
        if(responseMsg.IsSuccessStatusCode)
        {
            Console.WriteLine("Wow I just notified the users programmatically!");
        }
    }
}

Both approaches, manual via LCS and programmatic via RESTful API calls, will show users that a period of downtime is pending.

Begin the refresh
Depending on the size of your source environment, it might make sense to begin the refresh process immediately. The larger the source database, the longer it will take to copy to your target Azure SQL Database instance. While the copy is in progress, the target environment will still be online. The downtime will start after the copy is completed.

This process can be done manually via LCS. For the latest steps, see Refresh database. In a future release of LCS, you will also be able to trigger this process via a RESTful API.

**Reconfigure environment specific settings**

After the refresh is completed, use the **Sign off** button in LCS to close out of the operation. You can then start to configure the environment-specific settings.

First, sign in to the environment by using the admin account that can be found on the **Environment details** page in LCS. Here are typical areas of reconfiguration. You might require additional reconfiguration, based on your setup and the independent software vendor (ISV) solutions that are installed.

- **System administration > Setup > Batch groups**: Add the various Application Object Server (AOS) instances to the batch server groups that you require.
- **System administration > Setup > Entity Store**: Refresh the various entities that you require for Microsoft Power BI reporting.
- **System administration > Setup > System parameters**: Reconnect the environment to the LCS Help configuration for task guides.
- **System administration > Setup > Email > Email parameters**: Enter the Simple Mail Transfer Protocol (SMTP) settings if you use email in your UAT environment.
- **System administration > Inquiries > Batch jobs**: Select the jobs that you want to run in your UAT environment, and update the status to **Waiting**.

To complete this reconfiguration more quickly, we recommend that you build a custom web service endpoint that can be called on demand after the refresh is completed. An example of this type of web service will be added in a future update of this topic.

**Open the environment to users**

When the system is configured as you require, you can enable selected users to let them access the environment. By default, all users except the admin and Microsoft service accounts are disabled.

Go to **System administration > Users > Users**, and enable the users that should have access to the UAT environment. If many users must be enabled, you can complete this task more quickly by using the **Microsoft Excel Add-In**.
Database movement operations are a suite of self-service actions that can be used as part of data application lifecycle management (DataALM). This tutorial shows how to debug specific data and transactions from a recent copy of production data.

In this tutorial, you will learn how to:

- Refresh the user acceptance testing (UAT) environment.
- Add the IP address of your developer environment to an approved list ("safe list").
- Update your developer environment so that it connects to the UAT database.
- Set a breakpoint, and start to debug the data.

As an example of this scenario, a customer who has already gone live wants to debug a recent copy of production transactions from the development environment. In this way, the customer will be able to debug specific transactions that are stuck, or develop new features and reports by using realistic datasets.

**Prerequisites**

To do a refresh operation, you must have your production environment deployed, or you must have a minimum of two standard UAT environments. To complete this tutorial, you must have a developer environment deployed.

**IMPORTANT**

For debugging, we highly recommend that you use a DevTest environment that runs the same code and business logic that are available in your UAT environment. If you use multiple branches in version control, we recommend that the DevTest environment that is used to debug recent UAT or production transactions be connected to the same branch that you use to build packages for UAT and, later, for production. In this way, you don't have to run a database synchronization between your DevTest environment and UAT database, because the schema will be compatible.

Historically, this environment is known as a Hotfix/Support environment, because it's outside your usual code promotion path.

**Refresh the UAT environment**

This refresh operation overwrites the UAT environment with the latest copy of the production database. To complete this step, follow the instructions in Refresh for training purposes.

**Enable database access**

By default, all Sandbox Standard Acceptance Test environments use Microsoft Azure SQL Database as their database platform. The databases for these environments are protected by firewalls that restrict access to the Application Object Server (AOS) with which it was originally deployed.

To connect to the database, follow the instructions in Enable just-in-time access.

**NOTE**

Every time that a refresh is done, the firewall safe list is reset. You must add your DevTest environments back to this database when they are required in the future.
Update a OneBox DevTest environment to connect to the UAT database

In your developer environment, you must now update the web.config file to change the database connection. This step will let you run your local code and binaries that are configured against the database from UAT.

On your Services drive, go to the AoSService\WebRoot directory. (Typically, the Services drive is drive J or K.) Find the file that is named web.config, and make a backup of it. Then open the web.config file in Notepad or another editor, and find the following configurations:

- DataAccess.Database
- DataAccess.DBServer
- DataAccess.SqlPwd
- DataAccess.SqlUser

Update these configurations so that they use the values from the environment details page for the UAT environment in LCS.

```xml
<add key="DataAccess.Database" value="<example_axdb_fromAzure>" />
<add key="DataAccess.DBServer" value="<example_axdb_server.database.windows.net>" />
<add key="DataAccess.SqlPwd" value="<axdbadmin_password_from_LCS>" />
<add key="DataAccess.SqlUser" value="axdbadmin" />
```

Save the file. If you're operating in a cloud-hosted environment, run IISRESET. If you're on a Microsoft-managed developer machine and have limited permissions, make sure that Microsoft Visual Studio is closed.

Finally, open a web browser, go to the URL of your DevTest environment, and verify that you're pulling data from the UAT database.

Debug transactions in the DevTest environment

Now that your environment is correctly reconfigured, you can open Visual Studio and set a breakpoint in the application code that best meets your needs. Note that users in the UAT environment aren't affected while you debug in the DevTest environment.

Debugging batch

For scenarios where you need to debug batch jobs, on the debugging DevTest machine, you may need to restart the batch service before it will show up as an option to attach the debugger from Visual Studio. In addition, it may also be helpful to isolate this DevTest machine in its own batch group to ensure that any jobs that you want to debug will run on the DevTest machine.

Best practices

Here are some common best practices that will help guarantee that your debugging experience is quick, reliable, and doesn't disrupt other users in your organization:

- Make sure that the version of the code and binaries in the DevTest environment exactly match the version in the UAT environment. Connect the DevTest environment to the same branch that you build packages for deployment from. Alternatively, connect it to a "HotfixSupport" branch that is kept up to date with the latest customizations that are released.
- Don't run a database synchronization from Visual Studio. Otherwise, you will affect the availability of the schema in the UAT database and might affect users in the UAT environment.
- For the best experience, use a developer environment that was deployed in the same datacenter as the UAT environment.
Database movement operations are a suite of self-service actions that can be used as part of data application lifecycle management (DataALM). This tutorial shows how to export all the data and transactions from a sandbox standard user acceptance testing (UAT) environment.

In this tutorial, you will learn how to:

- Refresh the UAT environment.
- Run the export to the Asset library in Microsoft Dynamics Lifecycle Services (LCS).
- Download the database backup.
- Import the database, and prepare it so that it can be used in a developer environment.

As an example of this scenario, a customer who has already gone live wants to load a recent copy of production transactions into the development environment. In this way, the customer will be able to debug specific transactions, or develop new features and reports by using realistic datasets.

IMPORTANT

Database copy to a build environment is not supported. Learn more about build environments.

Known limitations

Because of recent restrictions by the Microsoft Azure SQL Database platform, we don’t recommend that you export your database if it’s larger than 200 gigabytes (GB). If you must export a larger database, we recommend that you use the legacy documentation until SQL Database can support larger exports. Note that this recommendation applies to export operations, not refresh operations. Refresh operations can support databases that are up to 4 terabytes (TB) in size.

Prerequisites

To do a refresh operation, you must have your production environment deployed, or you must have a minimum of two standard UAT environments. To complete this tutorial, you must have a developer environment deployed.

Refresh the UAT environment

This refresh operation overwrites the UAT environment with the latest copy of the production database. To complete this step, follow the instructions in Refresh for training purposes.

Back up to the Asset Library

From your sandbox Environment Details page, click the Maintain menu, and then select Move database.
A slider pane will open on the page where you can use the Export database action.
The environment will be unavailable for other servicing operations, such as Sandbox refresh or package deployment during this time. The source environment will be usable from a Dynamics user perspective.

After the export operation completes successfully, sign out of the servicing operation on your Environment details page. You can then see the asset in your Asset Library in the Database backups section.

---

The environment will be unavailable for other servicing operations, such as Sandbox refresh or package deployment during this time. The source environment will be usable from a Dynamics user perspective.

After the export operation completes successfully, sign out of the servicing operation on your Environment details page. You can then see the asset in your Asset Library in the Database backups section.
The .bacpac files are stored here and can be manually downloaded to your Tier 1 developer environments for import. In the future, Microsoft will provide APIs to trigger the export action, as well as list the available backup files in your asset library. This includes the secured URL for automatically downloading a backup asset file or copying it directly to your secure blob storage using Microsoft Azure Storage SDKs.

**Import the database**

After you've downloaded a database backup (.bacpac) file, you can begin the manual import operation on your Tier 1 environment. When you import the database, we recommend that you follow these guidelines:

- Keep a copy of the existing AxDB database, so that you can revert to it later if you must.
- Import the new database under a new name, such as AxDB_fromProd.

To ensure the best performance, copy the *.bacpac file to the local computer that you're importing from. Download sqlpackage .NET Core for Windows from Get sqlpackage .NET Core for Windows. Open a Command Prompt window, and run the following commands from the sqlpackage .NET Core folder.

```
SqlPackage.exe /a:import /sf:D:\Exportedbacpac\my.bacpac /tsn:localhost /tdn:<target database name> /p:CommandTimeout=1200
```

Here is an explanation of the parameters:

- **tsn (target server name)** – The name of the Microsoft SQL Server instance to import into.
- **tdn (target database name)** – The name of the database to import into. The database should not already exist.
- **sf (source file)** – The path and name of the file to import from.

**IMPORTANT**

To ensure that imported data is compatible with the metadata, you must trigger a full database synchronization from Visual Studio.

During import, the user name and password aren't required. By default, SQL Server uses Microsoft Windows authentication for the user who is currently signed in.

**Update the database**

Run the following SQL script against the imported database. This script adds back the users that you deleted from the source database and correctly links them to the SQL logins for this SQL Server instance. The script also turns change tracking back on. Remember to edit the final ALTER DATABASE statement so that it uses the name of your database.

```
CREATE USER axdeployuser FROM LOGIN axdeployuser
EXEC sp_addrolemember 'db_owner', 'axdeployuser'

CREATE USER axdbadmin FROM LOGIN axdbadmin
EXEC sp_addrolemember 'db_owner', 'axdbadmin'

CREATE USER axmrruntimeuser FROM LOGIN axmrruntimeuser
EXEC sp_addrolemember 'db_datareader', 'axmrruntimeuser'
EXEC sp_addrolemember 'db_datawriter', 'axmrruntimeuser'

CREATE USER axretaildatasyncuser FROM LOGIN axretaildatasyncuser
EXEC sp_addrolemember 'DataSyncUsersRole', 'axretaildatasyncuser'

CREATE USER axretailruntimeuser FROM LOGIN axretailruntimeuser
```
EXEC sp_addrolemember 'UsersRole', 'axretailruntimeuser'
EXEC sp_addrolemember 'ReportUsersRole', 'axretailruntimeuser'

EXEC sp_addrolemember 'DeployExtensibilityRole', 'axdeployextuser'

CREATE USER axdeployextuser FROM LOGIN axdeployextuser
EXEC sp_addrolemember 'DeployExtensibilityRole', 'axdeployextuser'

CREATE USER [NT AUTHORITY\NETWORK SERVICE] FROM LOGIN [NT AUTHORITY\NETWORK SERVICE]
EXEC sp_addrolemember 'db_owner', 'NT AUTHORITY\NETWORK SERVICE'

UPDATE T1
SET T1.storageproviderid = 0,
    T1.accessinformation = '','
    T1.modifiedby = 'Admin',
    T1.modifieddatetime = getdate()
FROM docuvalue T1
WHERE T1.storageproviderid = 1 --Azure storage

DROP PROCEDURE IF EXISTS SP_ConfigureTablesForChangeTracking
DROP PROCEDURE IF EXISTS SP_ConfigureTablesForChangeTracking_V2
GO

-- Begin Refresh Retail FullText Catalogs
DECLARE @RFTXNAME NVARCHAR(MAX);
DECLARE @RFTXSQL NVARCHAR(MAX);
DECLARE retail_ftx CURSOR FOR
SELECT OBJECT_SCHEMA_NAME(object_id) + '.' + OBJECT_NAME(object_id) fullname FROM SYS.FULLTEXT_INDEXES
    WHERE FULLTEXT_CATALOG_ID = (SELECT TOP 1 FULLTEXT_CATALOG_ID FROM SYS.FULLTEXT_CATALOGS WHERE NAME = 'COMMERCEFULLTEXTCATALOG');
OPEN retail_ftx;
FETCH NEXT FROM retail_ftx INTO @RFTXNAME;
BEGIN TRY
    WHILE @@FETCH_STATUS = 0
    BEGIN
        PRINT 'Refreshing Full Text Index ' + @RFTXNAME;
        EXEC SP_FULLTEXT_TABLE @RFTXNAME, 'activate';
        SET @RFTXSQL = 'ALTER FULLTEXT INDEX ON ' + @RFTXNAME + ' START FULL POPULATION';
        EXEC SP_EXECUTESQL @RFTXSQL;
        FETCH NEXT FROM retail_ftx INTO @RFTXNAME;
    END
END TRY
BEGIN CATCH
    PRINT error_message()
END CATCH
CLOSE retail_ftx;
DEALLOCATE retail_ftx;

-- End Refresh Retail FullText Catalogs

--Begin create retail channel database record--
declare @ExpectedDatabaseName nvarchar(64) = 'Default';
declare @DefaultDataGroupRecId BIGINT;
declare @ExpectedDatabaseRecId BIGINT;
IF NOT EXISTS (select 1 from RETAILCONNDATABASEPROFILE where NAME = @ExpectedDatabaseName)
BEGIN
    select @DefaultDataGroupRecId = RECID from RETAILCDXDATAGROUP where NAME = 'Default';
    insert into RETAILCONNDATABASEPROFILE (DATAGROUP, NAME, CONNECTIONSTRING, DATASTORETYPE)
        values (@DefaultDataGroupRecId, @ExpectedDatabaseName, NULL, 0);
    select @ExpectedDatabaseRecId = RECID from RETAILCONNDATABASEPROFILE where NAME = @ExpectedDatabaseName;
    insert into RETAILCDXDATASTORECHANNEL (CHANNEL, DATABASEPROFILE)
        select RCT.RECID, @ExpectedDatabaseRecId from RETAILCHANNELTABLE RCT
            inner join RETAILCHANNELTABLEEXT RCTEX on RCTEX.CHANNEL = RCT.RECID
    update RETAILCHANNELTABLEEXT set LIVECHANNELDATABASE = @ExpectedDatabaseRecId where LIVEXHANNELDATABASE = 0;
END;
--End create retail channel database record

Turn on change tracking
If change tracking was turned on in the source database, be sure to turn it on in the newly provisioned database in the target environment. To turn on change tracking, use the `ALTER DATABASE` command.

```
ALTER DATABASE [your database name] SET CHANGE_TRACKING = ON (CHANGE_RETENTION = 6 DAYS, AUTO_CLEANUP = ON);
```

To help guarantee that the current version of the store procedure that is related to change tracking is used in the new database, you must turn change tracking on or off for a data entity in Data management. You can choose any entity. This step is required in order to trigger a refresh of the store procedure.

**Start to use the new database**

To switch environments and use the new database, first stop the following services:

- World Wide Web Publishing Service
- Microsoft Dynamics 365 Unified Operations: Batch Management Service
- Management Reporter 2012 Process Service

After these services have been stopped, rename the AxDB database `AxDB_orig`, rename your newly imported database `AxDB`, and then restart the three services.

To switch back to the original database, reverse this process. In other words, stop the services, rename the databases, and then restart the services.

**Post steps for Commerce environments**

If you are using Commerce channels, when importing a database to a developer environment, which was originally exported from a self-service sandbox, the following additional steps must be performed on the destination developer environment. Without completing these steps, Commerce channels will not function.

1. To restore Commerce channels functionality, apply the latest Microsoft service update or quality update, which will create the channel database.
2. To restore any previously deployed channel database extensions, re-apply the corresponding Retail self-service deployable package.

**Reprovision the target environment**

```
IMPORTANT
Some environment-specific records are not included in automated database movement operations and require additional steps. These include the following:

- Commerce self-service installer references
- Commerce Scale Unit channel database configuration records
```

If you copy a database between environments, Commerce capabilities in the destination environment will not be fully functional until you perform the following additional steps.

**Initialize Commerce Scale Units**

If you are moving a database to a sandbox UAT or production environment, you must Initialize Commerce Scale Unit after the database movement operation is complete. The Commerce Scale Unit's association from the source environment will not copy over to the destination environment.

**Synchronize Commerce self-service installers**

To be able to access Commerce self-service installers in HQ, you must Synchronize self-service installers after the database movement operation is complete.
To run the Environment re-provisioning tool on the destination environment, run the following steps:

1. In your project's Asset Library, in the Software deployable packages section, select Import.
2. From the list of shared assets, select the Environment Reprovisioning Tool.
3. On the Environment details page for your destination environment, select Maintain > Apply updates.
4. Select the Environment Reprovisioning tool that you uploaded earlier, and then select Apply to apply the package.
5. Monitor the progress of the package deployment.

For more information about how to apply a deployable package, see Create deployable packages of models. For more information about how to manually apply a deployable package, see Install deployable packages from the command line.

**Re-activate POS devices**

If you use point of sale (POS) devices, after you import a database you must activate the POS devices again. Previously activated devices in the destination environment will no longer function. For more information, see Point of sale device activation.

**Reset the Financial Reporting database**

If you're using Financial Reporting, you must reset the Financial Reporting database by following the steps in Reset the Financial reporting data mart. (Financial Reporting was previously named Management Reporter.)

**Reenter data from encrypted and environment-specific fields in the target database**

In the client, enter the values that you documented for the encrypted and environment-specific fields. The following fields are affected. The field names are given in Table.Field format.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Where to Set the Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreditCardAccountSetup.SecureMerchantProperties</td>
<td>Select Accounts receivable &gt; Payments setup &gt; Payment services.</td>
</tr>
<tr>
<td>ExchangeRateProviderConfigurationDetails.Value</td>
<td>Select General ledger &gt; Currencies &gt; Configure exchange rate providers.</td>
</tr>
<tr>
<td>FiscalEstablishment_BR.ConsumerEFDocCsc</td>
<td>Select Organization administration &gt; Fiscal establishments &gt; Fiscal establishments.</td>
</tr>
<tr>
<td>FiscalEstablishmentStaging.CSC</td>
<td>This field is used by the Data Import/Export Framework (DIXF).</td>
</tr>
<tr>
<td>HcmPersonIdentificationNumberPersonIdentificationNumber</td>
<td>Select Human resources &gt; Workers &gt; Workers. On the Worker tab, in the Personal information group, select Identification numbers.</td>
</tr>
</tbody>
</table>
### Community tools

Are you looking for more tools to help you import backup files into your developer environments? Here are some other sources of information:

- [D365fo.Tools](#) provides many valuable tools that have been created by the community.
- [Community-provided open source projects on GitHub](#).

### Known issues

**The export database is in a "Preparation failed" state**

If the automation from LCS times out, the state of the export database is changed to [Preparation failed](#). The export operation to export to the Asset library is still running in SQL Database. To resolve this issue, you can use the [Resume](#) button to reconnect the process with SQL Database. The process should then be successfully completed.

**The export database takes a very long time**

The Azure SQL team recently announced that the [Import/Export](#) application programming interface (API) that LCS uses has variable execution times for any database that is over 200 GB in size. If you encounter this issue, you can either [connect your DevTest environment directly to the UAT database](#) or follow the [legacy documentation](#). We don't recommend that you export databases for backup purposes, because the point-in-time restore functionality is available and included with your environment.

The Lifecycle Services team is working directly with the Azure SQL team to increase the performance of the Import/Export API and will make improvements in upcoming releases of LCS.

**I can't download Management Studio installation files**

When you try to download the Microsoft SQL Server Management Studio installer, you might receive the following error message:

| Your current security settings do not allow this file to be downloaded. |

To work around this issue, follow these steps to enable file downloads.

1. In your web browser, open [Internet options](#).
2. On the [Security](#) tab, select the [Internet](#) zone, and then select [Custom level](#).
3. Scroll to [Downloads](#), and then, under [File download](#), select the [Enable](#) option.

**Database synchronization fails**
When you sync the database against the newly imported database from Microsoft Visual Studio, the synchronization might fail, and you might receive the following error message:

```
Failed to open SQL connection syncengine.exe exited with code -1.
```

In this case, the following message is also logged under event ID 140 in the Windows application log:

```
Object Server Database Synchronizer: The internal system table version number stored in the database is higher than the version supported by the kernel (141/138). Use a newer Microsoft Dynamics kernel, or start Microsoft Dynamics using the -REPAIR command line parameter to enforce synchronization.
```

This issue can occur when the platform build number of the current environment is lower than the platform build number of the source environment. To resolve the issue, follow one of these steps, depending on your circumstances:

- Use the Updates tiles on the environment page in LCS to upgrade the platform in the current environment so that it matches the platform in the source environment.

- Run the following query to adjust the expected version in the database.

```
UPDATE SQLSYSTEMVARIABLES
SET VALUE = 138
WHERE PARM = 'SYSTABVERSION'
```

**NOTE**
The value `138` in this query is taken from the event log message, where version 138 was expected in this particular environment.

**Performance**
The following guidelines can help you achieve optimal performance:

- Always import the .bacpac file locally on the computer that runs the SQL Server instance. Don’t import it from Management Studio on a remote machine.

- In a one-box environment that is hosted in Azure, put the .bacpac file on drive D when you import it. (A one-box environment is also known as a Tier 1 environment.) For more information about the temporary drive on Azure virtual machines (VMs), see the Understanding the temporary drive on Windows Azure Virtual Machines blog post.

- Grant the account that runs the SQL Server Windows service `Instance File Initialization` rights. In this way, you can help improve the speed of the import process and the speed of a restore from a *.bak file. For a developer environment, you can easily make sure that the account that runs the SQL Server service has these rights by setting SQL Server to run as the `axlocaladmin` account.
Database movement operations are a suite of self-service actions that can be used as part of data application lifecycle management (DataALM). “Golden configuration” refers to a common practice among customers and partners in the Microsoft Dynamics ecosystem, where a developer environment is used as a configuration store. In this way, implementation projects can store finalized global and company-specific settings in a database that can later become a baseline for Conference Room Pilots, mock go-lives, and go-lives. This tutorial shows how to prepare a golden configuration database and hydrate a target user acceptance testing (UAT) environment.

In this tutorial, you will learn how to:

- Prepare the golden configuration database for Microsoft Azure SQL Database.
- Run the import to the target UAT environment.
- Copy the UAT environment into a production environment.

As an example of this scenario, a customer who hasn't gone live is instead preparing for a Conference Room Pilot, mock go-live, or go-live. This scenario supports promoting a baseline golden database from a developer environment to a UAT environment and eventually to production.

Prerequisites

To complete this tutorial, you must have a developer environment that is deployed with a database that is curated as a golden configuration database. You must also have at least one standard UAT environment deployed and, optionally, a production environment.

The developer environment must run the same application version as the target UAT environment. In addition, the platform version of the developer environment must be earlier than or the same as the platform version in the target UAT environment.

Before you begin

**Supported SQL Server collation**

The only supported collation databases in the cloud is `SQL_Latin1_General_CP1_CI_AS`. Make sure that your Microsoft SQL Server and database collations in development environments are set to this value. Also make sure that any configuration environments that are published to sandbox have this collation.

**Document the values of encrypted fields**

Encrypted and environment-specific values can't be imported into a new environment. After you’ve completed the import, you must reenter some data from your source environment in your target environment.

Because of a technical limitation that is related to the certificate that is used for data encryption, values that are stored in encrypted fields in a database will be unreadable after that database is imported into a new environment. Therefore, after an import, you must manually delete and reenter values that are stored in encrypted fields. New values that are entered in encrypted fields after an import will be readable. The following fields are affected. The field names are given in `Table.Field` format.

<table>
<thead>
<tr>
<th>FIELD NAME</th>
<th>WHERE TO SET THE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD NAME</td>
<td>WHERE TO SET THE VALUE</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>CreditCardAccountSetup.SecureMerchantProperties</td>
<td>Select Accounts receivable &gt; Payments setup &gt; Payment services.</td>
</tr>
<tr>
<td>ExchangeRateProviderConfigurationDetails.Value</td>
<td>Select General ledger &gt; Currencies &gt; Configure exchange rate providers.</td>
</tr>
<tr>
<td>FiscalEstablishment_BR.ConsumerEDocCsc</td>
<td>Select Organization administration &gt; Fiscal establishments &gt; Fiscal establishments.</td>
</tr>
<tr>
<td>FiscalEstablishmentStaging.CSC</td>
<td>This field is used by the Data Import/Export Framework (DIXF).</td>
</tr>
<tr>
<td>HcmPersonIdentificationNumber.PersonIdentificationNumber</td>
<td>Select Human resources &gt; Workers &gt; Workers. On the Worker tab, in the Personal information group, select Identification numbers.</td>
</tr>
<tr>
<td>HcmWorkerActionHire.PersonIdentificationNumber</td>
<td>This field has been obsolete since Microsoft Dynamics AX 7.0 (February 2016). It previously appeared on the All worker actions page (Human resources &gt; Workers &gt; Actions &gt; All worker actions).</td>
</tr>
<tr>
<td>SysEmailSMPTPassword.Password</td>
<td>Select System administration &gt; Email &gt; Email parameters.</td>
</tr>
<tr>
<td>SysOAuthUserTokens.EncryptedAccessToken</td>
<td>This field is used internally by Application Object Server (AOS). It can be ignored.</td>
</tr>
<tr>
<td>SysOAuthUserTokens.EncryptedRefreshToken</td>
<td>This field is used internally by AOS. It can be ignored.</td>
</tr>
</tbody>
</table>

**If you’re running Commerce components, document encrypted and environment-specific values**

The values on the following pages are either environment-specific or encrypted in the database. Therefore, all the imported values will be incorrect.

- Payments services *(Accounts receivable > Payments setup > Payments services)*
- Hardware profiles *(Retail and commerce > Channel setup > POS setup > POS profiles > Hardware profiles)*

**Create a copy of the source database**

Back up the source database using SSMS. Right-click the source database, and select Tasks > Backup option. After this is completed, right-click the Databases folder in the SSMS navigation window, and select Restore database. Choose the backup that you just made, but give the target database a new name such as AXDB_CopyForExport.

**Prepare the database**

Run the following script against the AxDB_CopyForExport database that you created in the previous section. This script makes the following changes:

- Set the **SysGlobalConfiguration** flag to inform the application that the database is Azure-based.
- Remove a reference to tempDB in the XU_DisableEnableNonClusteredIndexes procedure. References to tempDB aren’t allowed in an Azure SQL database. The database synchronization process will re-create the reference later.
- Drop users, because Microsoft Windows users are forbidden in Azure SQL databases. Other users must be re-created later, so that they're correctly linked to the appropriate sign-in on the target server.
- Clear encrypted hardware profile merchant properties.

A successful export and import of the database requires all these changes.

```sql
update sysglobalconfiguration
set value = 'SQLAZURE'
where name = 'BACKENDDB'

update sysglobalconfiguration
set value = 1
where name = 'TEMPTABLEINAXDB'

drop procedure if exists XU_DisableEnableNonClusteredIndexes
drop procedure if exists SP_ConfigureTablesForChangeTracking
drop procedure if exists SP_ConfigureTablesForChangeTracking_V2
drop schema [NT AUTHORITY\NETWORK SERVICE]
drop user [NT AUTHORITY\NETWORK SERVICE]
drop user axdbadmin
drop user axdeployuser
drop user axmruntimeuser
drop user axretaildatasyncuser
drop user axretailruntimeuser
drop user axdeployextuser

--Tidy up the batch server config from the previous environment
DELETE FROM SYSSERVERCONFIG

--Tidy up server sessions from the previous environment
DELETE FROM SYSSERVERSESSIONS

--Tidy up printers from the previous environment
DELETE FROM SYSCORPNETPRINTERS

--Tidy up client sessions from the previous environment
DELETE FROM SYSCLIENETSESSIONS

--Tidy up batch sessions from the previous environment
DELETE FROM BATCHSERVERCONFIG

--Tidy up batch server to batch group relation table
DELETE FROM BATCHSERVERGROUP

-- Clear encrypted hardware profile merchant properties
update dbo.RETAILHARDWAREPROFILE set SECUREMERCHANTPROPERTIES = null where SECUREMERCHANTPROPERTIES is not null
```

Export the database from SQL Server

Open a Command Prompt window, and run the following commands.

**IMPORTANT**

The 140 folder reflects the current version. You must use the version that is available in your sandbox environment. Therefore, you might have to install the latest version of Microsoft SQL Server Management Studio in your development environment.
Here is an explanation of the parameters:

- **ssn (source server name)** – The name of the SQL Server to export from. For the purposes of this topic, the name should always be **localhost**.
- **sdn (source database name)** – The name of the database to export.
- **tf (target file)** – The path and name of the file to export to. The folder should already exist, but the export process will create the file.

### Import the database

Upload the .bacpac file that was created in the previous step to the Database backup section in your LCS project's Asset Library. Then begin the import. The target UAT environment’s databases will be overwritten by the golden configuration database.

**NOTE**

Certain elements are not copied as part of the import database step. In the golden configuration scenario, this would impact things such as Email Addresses and Print Management setup. These settings ideally should be populated as part of the master data migration in the steps below, and should not be part of the golden configuration database.

To import a database that is prepared from a developer environment to a standard user acceptance test (UAT), or a database previously exported from a UAT environment, follow the steps outlined below:

1. Go to your target sandbox Environment Details page, and select the Maintain > Move database menu option.
2. Select Import database and choose your source database backup (.bacpac format) file from the Asset Library.
3. Note the warnings. Review the list of data elements that are cleaned up from the backup file.
4. The import operation will begin immediately.

**NOTE**

All users except the Admin user and other internal service user accounts will be unavailable after import. Therefore, the Admin user can delete or obfuscate data before other users are allowed back into the system.

To import a database to a developer environment after you've downloaded a database backup (.bacpac) file, you can begin the manual import operation on your Tier 1 environment. When you import the database, we recommend that you follow these guidelines:

- Keep a copy of the existing AxDB database, so that you can revert to it later if needed.
- Import the new database under a new name, such as AxDB_fromProd.

To ensure the best performance, copy the *.bacpac file to the local computer that you're importing from. Download sqlpackage .NET Core for Windows from Get sqlpackage .NET Core for Windows. Open a Command Prompt window, and run the following commands from the sqlpackage .NET Core folder.
SqlPackage.exe /a:import /sf:D:\Exportedbacpac\my.bacpac /tsn:localhost /tdn:<target database name> /p:CommandTimeout=1200

Here is an explanation of the parameters:

- **tsn (target server name)** – The name of the Microsoft SQL Server instance to import into.
- **tdn (target database name)** – The name of the database to import into. The database should not already exist.
- **sf (source file)** – The path and name of the file to import from.

**NOTE**

During import, the user name and password aren't required. By default, SQL Server uses Microsoft Windows authentication for the user who is currently signed in.

For information about how to complete the manual import operations into a Tier 1 environment, see Import the database.

**Perform master data migration**

Now that the UAT environment is hydrated with the golden configuration, you can begin to migrate master data. You can do this data migration by using data entities. We recommend that you complete your data migration activities before you copy the UAT environment to production, because you will have access to the database in the UAT environment for troubleshooting.

**IMPORTANT**

Files stored in Azure blob storage are not copied from UAT to Production in the next step. This includes document attachments and custom Microsoft Office templates. If your go-live requires attachments or custom templates, you will want to import those in the Production environment directly.

**Copy the sandbox database to production**

When you’re ready to do a mock go-live or actual go-live, you can copy the UAT environment to production. This process is often referred to as cutover. We recommend that you do a cutover more than one time before your actual go-live. In this way, you can get detailed time estimates for each step of the process.

Determine the Environment type of your production environment and follow the relevant steps accordingly.

**Self-service**

1. In LCS, open the Full details for the production environment to load the Environment page.
2. In the Maintain menu, select Move database.
3. For the operations options, select Refresh database.
4. In the Source environment, select the sandbox where your golden configuration is. Note the important instructions found on the Refresh database page for this operation.
5. Select the check box to confirm that you understand this operation will overwrite the production database.

   The operation starts immediately after submitting the request.

**Microsoft-managed**

1. In LCS, on the project home page, select Service requests.
2. On the Service requests page, select Add, and then select Sandbox to Production.
3. In the **Sandbox to Production** dialog box, follow these steps:

   a. In the **Source environment name** field, select the sandbox environment to copy the database from.
   b. Set the **Preferred downtime start date** and **Preferred downtime end date** fields. The end date must be at least four hours after the start date. To help ensure that resources are available to run the request, it's recommended that you submit your request at least 24 hours before your preferred downtime window.
   c. Select the check boxes at the bottom to agree to the terms.

**IMPORTANT**

Every database refresh will create a new database that will reset the **Point-in-time-restore** chain of restore points.

---

**Reconfigure environment specific settings**

After the refresh is completed, use the **Sign off** button in LCS to close out of the operation. You then can start to configure the environment-specific settings.

First, sign in to the environment by using the admin account that can be found on the **Environment details** page in LCS. Here are some typical areas of reconfiguration. You might require additional reconfiguration, based on your setup and the independent software vendor (ISV) solutions that are installed:

- **System administration > Setup > Batch groups**: Add the various AOS instances to the batch server groups that you require.
- **System administration > Setup > Entity Store**: Update the various entities that you require for Microsoft Power BI reporting.
- **System administration > Setup > System parameters**: Reconnect the environment to the LCS Help configuration for task guides.
- **System administration > Setup > Email > Email parameters**: Enter the Simple Mail Transfer Protocol (SMTP) settings if you use email in your UAT environment.
- **System administration > Setup > Integration configuration > Azure storage account connection string**: Enter the storage account string.
- **System administration > Setup > System Parameters**: On the **Document connections** tab enter the Azure Key and Application Secret.
- **System administration > Inquiries > Batch jobs**: Select the jobs that you want to run in your UAT environment, and update the status to **Waiting**.

**NOTE**

As a best practice, all mission-critical batch jobs that will run with recurrence should be created and run by the admin account. The admin should be a generic user such as **erp@customer.com**. It should not be linked to a specific employee's Azure Active Directory (Azure AD) account, because that account might be disabled later if the employee leaves the company.

---

**Open the environment to users**

When the system is configured as you require, you can enable selected users to access the environment. By default, all users except the admin and Microsoft service accounts are disabled.

Go to **System administration > Users > Users**, and enable the users that should have access to the Production environment. If many users must be enabled, you can complete this task more quickly by using the Microsoft Excel Add-In.
Community tools

Are you looking for more tools to help you prepare backup files from your developer environments? Here are some other sources of information:

- **D365fo.Tools** provides many valuable tools that are created by the community.
- **Community-provided open source projects on GitHub.**
Database movement operations are a suite of self-service actions that can be used as part of data application lifecycle management (DataALM). In some situations, destructive testing must be done on an environment. In this context, destructive testing means that the environment is rendered no longer useful for continued testing. Destructive testing is typical in an implementation lifecycle during Conference Room Pilots. This tutorial shows how to use database movement operations to facilitate destructive testing.

In this tutorial, you will learn two approaches:

- Use a database backup asset.
- Use point-in-time restore.

As an example of this scenario, a customer wants to do a Conference Room Pilot and wants to start with an environment that has no transactions (that is, no sales orders or purchase orders). The customer will be traveling from physical warehouse to physical warehouse throughout the geographic region to do the same pilot, and wants the environment to be "reset" before each pilot is done.

Prerequisites

To complete this tutorial, you must have a standard user acceptance testing (UAT) environment deployed in your project.

Using a database backup

If you've prepared a database backup (.bacpac) file that is already at the starting point for the test, the easiest approach is to upload the backup file to the Database backup section in your LCS project's Asset Library. It can then be imported to your target environment as described here.

To import a database that is prepared from a developer environment to a standard user acceptance test (UAT), or a database previously exported from a UAT environment, follow the steps outlined below:

1. Go to your target sandbox Environment Details page, and select the Maintain > Move database menu option.
2. Select Import database and choose your source database backup (.bacpac format) file from the Asset Library.
3. Note the warnings. Review the list of data elements that are cleaned up from the backup file.
4. The import operation will begin immediately.

NOTE

All users except the Admin user and other internal service user accounts will be unavailable after import. Therefore, the Admin user can delete or obfuscate data before other users are allowed back into the system.

To import a database to a developer environment after you've downloaded a database backup (.bacpac) file, you can begin the manual import operation on your Tier 1 environment. When you import the database, we recommend that you follow these guidelines:

- Keep a copy of the existing AxDB database, so that you can revert to it later if needed.
- Import the new database under a new name, such as AxDB_fromProd.
To ensure the best performance, copy the *.bacpac file to the local computer that you're importing from. Download sqlpackage .NET Core for Windows from Get sqlpackage .NET Core for Windows. Open a Command Prompt window, and run the following commands from the sqlpackage .NET Core folder.

```sql
SqlPackage.exe /a:import /sf:D:\Exportedbacpac\my.bacpac /tsn:localhost /tdn:<target database name> /p:CommandTimeout=1200
```

Here is an explanation of the parameters:

- **tsn (target server name)** – The name of the Microsoft SQL Server instance to import into.
- **tdn (target database name)** – The name of the database to import into. The database should not already exist.
- **sf (source file)** – The path and name of the file to import from.

**NOTE**

During import, the user name and password aren't required. By default, SQL Server uses Microsoft Windows authentication for the user who is currently signed in.

For information about how to complete the manual import operations into a Tier 1 environment, see Import the database.

**Database backup pros and cons**

The advantage of using backup file assets is that you can keep importing the same file to get back to the starting point for the test.

The disadvantage is that if many configurations (for example, batch jobs) must be set after the import is completed but before users can begin, more effort will be required before each destructive testing session.

**Using point-in-time restore**

If you didn't start with a database backup (.bacpac) file but instead have the UAT environment in a known good state, you can just record the date and time in your time zone. You can then begin the destructive testing. Then, when the testing is completed, you can restore the environment to the previous time by using the following steps.

To restore the database of a standard user acceptance test (UAT) environment to a previous point-in-time, follow the steps outlined below:

1. Go to your target sandbox Environment Details page, and select the Maintain > Move database menu option.
2. Select the Point-in-time restore option and choose a point-in-time.
3. Note the warnings. Review the list of data elements that are not copied over from the previous point-in-time.
4. The restore operation will begin immediately.

**Point-in-time restore pros and cons**

The advantage of using point-in-time-restore is that you can avoid dealing with database backup (.bacpac) files and can reduce the time between destructive testing sessions.

The disadvantage is that, because of current limitations of point-in-time restore, you must record a new restore date and time in your time zone after the restore is completed. Because point-in-time restore always creates a new database, the original date and time that were used won't be available as a restore point on the new database.
The Database Movement application programming interface (API) is a RESTful endpoint that is used to manage the data lifecycle of Microsoft Dynamics 365 environments. It provides a versioned set of capabilities that you can currently use to copy databases between environments, and to list and download database backups. More supported actions will be added in later releases.

What is supported by the Database Movement API?

The Database Movement API exposes RESTful endpoints for the following Dynamics 365 services:

- Dynamics 365 Finance
- Dynamics 365 Supply Chain Management
- Dynamics 365 Commerce

Next steps

- Learn how to set up authentication.
- Review the API reference.
This topic provides an overview of the versioning and breaking change policies for the Database Movement application programming interface (API).

Support and deprecation information

As new versions of the REST APIs are released, earlier versions will be retired. Microsoft will declare a version deprecated at least six months before it retires an API endpoint.

By incrementing the version number of the API (for example, from v1 to v2), Microsoft announces that the lowest version (in this example, v1) is immediately deprecated and will no longer be supported six months after the announcement. However, Microsoft might make exceptions to this policy for service health and security issues.

When an API is marked as deprecated, a date value will be entered in the VersionEOL (Version end of life) field. Therefore, you can proactively monitor this field and plan for upcoming changes.

Compatible and breaking changes

Microsoft will provide details of API changes in the private preview group. If the changes are non-breaking in nature, the API version number will remain the same. If the changes are breaking in nature, Microsoft will increment the API version number.

Here are some examples of breaking changes:

- The URL or fundamental request/response is changed.
- A declared property is removed or renamed, or its type is changed.
- The API or API parameters are removed or renamed.
- A required request parameter is added.

Here are some examples of non-breaking changes:

- Properties are added that are nullable or have a default value.
- A member is added to an enumeration.
- Paging is introduced to existing collections.
- Error codes are changed.
- The order of properties in requests or responses is changed.

Example of VersionEOL in a response contract

The following example shows a response contract in JavaScript Object Notation (JSON) format. All response contracts contain the VersionEOL property of which has a default value of DateMax() from the Microsoft .NET Framework. Your applications can monitor the value of this field on responses from Microsoft to get an immediate alert when Microsoft has deprecated a specific endpoint or a whole API version.

```json
{
  "IsSuccess": true,
  "OperationActivityId": "55eb4327-9346-4c7b-82bd-fe8ef15112c6",
  "ErrorMessage": null,
  "VersionEOL": "9999-12-31T23:59:59.9999999"
}
```
This topic provides overview information about how to authenticate with the Database Movement application programming interface (API).

**Fundamentals**

To call the Database Movement API, your application must acquire an access token from the Microsoft identity platform. The access token contains information about your application and the permission that it has to call resources in Microsoft Dynamics Lifecycle Services (LCS).

**Access token**

Access tokens that are issued by the Microsoft identity platform are base64–encoded JavaScript Object Notation (JSON) Web Tokens (JWTs). They contain information (claims) that the Database Movement API and other web APIs that are secured by the Microsoft identity platform use to validate the caller and make sure that the caller has the correct permissions to perform the operation that they are requesting. During calls, you can treat access tokens as opaque. You should always transmit access tokens over a secure channel, such as Transport Layer Security (TLS) and Hypertext Transfer Protocol Secure (HTTPS).

Here is an example of an access token that is issued by the Microsoft identity platform.

```
EwAoA8l6BAAU7pQPdpl/D7xLwCtG3SlyQAAxU7IAY94aAS4OK5xQ/SUS9H2Z0X5s19kqLpoC9QVF8lnpYXH7NdJ6uWw+1jIfVNTewd2z42VPvqooMWrUXs1iGn1A7e1k4v43FbKUeX6U8n19n5JHkL6v48nAaNPTTJ86YqOqT7x5b loses3CAcGqPeURLV76Y4BNMtx18wek418kn3p111m5JXOHSY57G0

http://lcsapi.lcs.dynamics.com/databasemovement/v1/databases
```

To call the Database Movement API, you attach the access token as a bearer token to the authorization header in your HTTP request. Here is an example.

```
HTTP/1.1
Authorization: Bearer EwAoA8l6BAAU7pQPdpl/D7xLwCtG3SlyQAAxU7IAY94aAS4OK5xQ/SUS9H2Z0X5s19kqLpoC9QVF8lnpYXH7NdJ6uWw+1jIfVNTewd2z42VPvqooMWrUXs1iGn1A7e1k4v43FbKUeX6U8n19n5JHkL6v48nAaNPTTJ86YqOqT7x5b loses3CAcGqPeURLV76Y4BNMtx18wek418kn3p111m5JXOHSY57G0
```

Register a new application by using the Azure portal

1. Sign in to the Microsoft Azure portal by using a work or school account, or a personal Microsoft account.

2. If your account gives you access to more than one tenant, select your account in the upper-right corner, and set your portal session to the Azure Active Directory (Azure AD) tenant that you want.

3. In the left pane, select the Azure Active Directory service, and then select App registrations > New registration.

4. When the Register an application page appears, enter your application's registration information:
**Name** – Enter a meaningful application name that will be shown to users of the app.

**Supported account types** – Select the types of accounts that your app should support.

<table>
<thead>
<tr>
<th>SUPPORTED ACCOUNT TYPES</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| Accounts in this organizational directory only | Select this option if you're building a line-of-business app. This option isn't available unless you're registering the app in a directory. This option is mapped to **Azure AD only single-tenant**.

This option is the default option unless you're registering the app outside a directory. In that case, the default option is **Azure AD multi-tenant and personal Microsoft accounts**. |
| Accounts in any organizational directory | Select this option to target all business and educational customers. This option is mapped to **Azure AD only multi-tenant**.

If you registered the app as **Azure AD only single-tenant**, you can use the Authentication blade to update it to **Azure AD only multi-tenant** and then back to **Azure AD only single-tenant**. |
| Accounts in any organizational directory and personal Microsoft accounts | Select this option to target the widest set of customers. This option is mapped to **Azure AD multi-tenant and personal Microsoft accounts**.

If you registered the app as **Azure AD multi-tenant and personal Microsoft accounts**, you can't change this setting in the user interface (UI). Instead, you must use the application manifest editor to change the supported account types. |

**Redirect URI (optional)** – Select the type of app that you're building: **Web** or **Public client (mobile & desktop)**. Then enter the redirect URI (or reply URL) for the app.

- For web apps, provide the base URL of the app. For example, `http://localhost:31544` might be the URL for a web app that runs on your local machine. Users then use this URL to sign in to a web client app.
- For public client apps, provide the URI that Azure AD uses to return token responses. Enter a value that is specific to your app, such as `myapp://auth`.

To see specific examples for web apps or native apps, see the quick start guides from Azure AD.

5. Under **API permissions**, select **Add a permission**. Then, on the **APIs my organization uses** tab, search for **Dynamics Lifecycle services**, and add the **user_impersonation** permission to your app.

6. Select **Register**.
Azure AD assigns a unique application ID (client ID) to your app, and you're taken to the **Overview** page for your app. To add more capabilities to your app, you can select other configuration options, such as options for branding, and for certificates and secrets.
This topic provides an overview of throttling for the Database Movement application programming interface (API).

**Rate limits**

To help maintain the reliability of the service and reduce costs, throttling will be turned on for the Database Movement API. Throttling helps protect against malicious and excessive use of the RESTful endpoints. Database movement operations are some of the most time-consuming and CPU-intensive tasks that can be run from Microsoft Dynamics Lifecycle Services (LCS), and Microsoft might change the current call limits later.

**Current limits**

Currently, the Database Movement API has a global call limit of **three executions per 24-hour timeframe** for all actions that trigger a new operation in LCS. These operations include database refresh operations.

If you exceed the limit, you won't be able to start a new operation and will receive an error that resembles the following example.

```json
{
    "IsSuccess": false,
    "OperationActivityId": "55eb4327-9346-4c7b-82bd-fe8ef15112c6",
    "ErrorMessage": "Maximum allowed API operations are 3 from 2019-09-30T04:01:01.9999999",
    "VersionEOL": "9999-12-31T23:59:59.9999999"
}
```
Welcome to the Database Movement application programming interfaces (API) reference for the version 1 (v1) endpoint. Use the table of contents to the left to view details of each endpoint that is available.
You can retrieve a list of database backups from the Project asset library.

**Permissions**

One of the following permissions is required to call this application programming interface (API). For more information about permissions and how to select them, see Authentication.

<table>
<thead>
<tr>
<th>PERMISSION TYPE</th>
<th>PERMISSIONS (FROM LEAST PRIVILEGED TO MOST PRIVILEGED)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delegated (work or school account)</td>
<td>user_impersonation</td>
</tr>
</tbody>
</table>

**HTTP request**

GET /databasemovement/v1/databases/project/{projectId}

**Request headers**

<table>
<thead>
<tr>
<th>HEADER</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization</td>
<td>Bearer (token) (required)</td>
</tr>
<tr>
<td>'x-ms-version'</td>
<td>'2017-09-15' (required)</td>
</tr>
<tr>
<td>Content-Type</td>
<td>application/json</td>
</tr>
</tbody>
</table>

**Request body**

Don't supply a request body for this method.

**Response**

The response is always a 200 OK response, unless you aren't correctly authenticated. Be sure to use the IsSuccess property to evaluate the success or failure of the action.

**Example**

GET /databasemovement/v1/databases/project/12345
{ "DatabaseAssets": [
  {
    "Id": "4a2c52d4-49ca-4606-94a9-92b3a6b42985",
    "ProjectId": 12345,
    "OrganizationId": 1,
    "Name": "LanesBackupRecord",
    "FileName": "RandomFile.bacpac",
    "FileDescription": "",
    "FileLocation": "https://scrubbed.blob.core.windows.net/product-ax7productname/e6244b15-5112-4d1d-a422-49c63496ab6d/AX7ProductName-12-17-83c6e642-676a-4048-b413-6e284f5d1f55-e6244b15-5112-4d1d-a422-49c63496ab6d?sv=2015-12-11&sr=b&sig=rO3zmAZ3zM6s%2FV2BelihBA2LMVqMxsxbtsnbauvd8keYo3D&se=2019-09-28T15%3A18%3A05Z&sp=r",
    "ModifiedDateTime": "2019-09-27T15:17:35.867",
    "CreatedDateTime": "2019-09-27T15:17:35.867",
    "CreatedByName": null,
    "ModifiedBy_Name": null
  }
] },
You can create a database refresh between two environments. Note that the same validation rules from the details page in Microsoft Dynamics Lifecycle Services (LCS) apply to the application programming interface (API).

Permissions

One of the following permissions is required to call this API. For more information about permissions and how to select them, see Authentication.

<table>
<thead>
<tr>
<th>PERMISSION TYPE</th>
<th>PERMISSIONS (FROM LEAST PRIVILEGED TO MOST PRIVILEGED)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delegated (work or school account)</td>
<td>user_impersonation</td>
</tr>
</tbody>
</table>

HTTP request

**POST**

/databasemovement/v1/refresh/project/{projectId}/source/{sourceEnvironmentId}/target/{targetEnvironmentId}

Request headers

<table>
<thead>
<tr>
<th>HEADER</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization</td>
<td>Bearer (token) (required)</td>
</tr>
<tr>
<td>'x-ms-version'</td>
<td>'2017-09-15' (required)</td>
</tr>
<tr>
<td>Content-Type</td>
<td>application/json</td>
</tr>
</tbody>
</table>

Request body

Don't supply a request body for this method.

Response

The response is always a **200 OK** response, unless you aren't correctly authenticated. Be sure to use the IsSuccess property to evaluate the success or failure of the action.

Example

**POST** /databasemovement/v1/refresh/project/12345/source/5362377c-bc37-4f92-b30e-fe0c1e664cc8/target/6a90b45f-1764-4077-b924-3f4671548237
{  "IsSuccess": true,
  "OperationActivityId": "55eb4327-9346-4c7b-82bd-fe8ef15112c6",
  "ErrorMessage": null,
  "VersionEOL": "9999-12-31T23:59:59.9999999"
}
Create a database export

11/24/2021 • 2 minutes to read • Edit Online

You can create a database export from a sandbox environment to the project's asset library. Note that the same validation rules from the details page in Microsoft Dynamics Lifecycle Services (LCS) apply to the application programming interface (API).

Permissions

One of the following permissions is required to call this API. For more information about permissions and how to select them, see Authentication.

<table>
<thead>
<tr>
<th>PERMISSION TYPE</th>
<th>PERMISSIONS (FROM LEAST PRIVILEGED TO MOST PRIVILEGED)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delegated (work or school account)</td>
<td>user_impersonation</td>
</tr>
</tbody>
</table>

HTTP request

POST /databasemovement/v1/export/project/{projectId}/environment/{environmentId}/backupName/{backupName}

Request headers

<table>
<thead>
<tr>
<th>HEADER</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization</td>
<td>Bearer (token) (required)</td>
</tr>
<tr>
<td>Content-Type</td>
<td>application/json</td>
</tr>
</tbody>
</table>

Request body

Don't supply a request body for this method.

Response

The response is always a 200 OK response, unless you aren't correctly authenticated. Be sure to use the IsSuccess property to evaluate the success or failure of the action.

Example

POST /databasemovement/v1/export/project/12345/environment/5362377c-bc37-4f92-b30e-fe0c1e664cc0/backupName/TestBackupViaAPI
{"IsSuccess": true,
"OperationActivityId": "55eb4327-9346-4c7b-82bd-fe8ef15112c6",
"ErrorMessage": null,
"VersionEOL": "9999-12-31T23:59:59.9999999"}
You can get the status of an ongoing operation.

Permissions

One of the following permissions is required to call this application programming interface (API). For more information about permissions and how to select them, see Authentication.

<table>
<thead>
<tr>
<th>PERMISSION TYPE</th>
<th>PERMISSIONS (FROM LEAST PRIVILEGED TO MOST PRIVILEGED)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delegated (work or school account)</td>
<td>user_impersonation</td>
</tr>
</tbody>
</table>

HTTP request

GET /databasemovement/v1/fetchstatus/project/{projectId}/environment/{environmentId}/operationactivity/{operationactivityId}

Request headers

<table>
<thead>
<tr>
<th>HEADER</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization</td>
<td>Bearer (token) (required)</td>
</tr>
<tr>
<td>'x-ms-version'</td>
<td>'2017-09-15' (required)</td>
</tr>
<tr>
<td>Content-Type</td>
<td>application/json</td>
</tr>
</tbody>
</table>

Request body

Don't supply a request body for this method.

Response

The response is always a 200 OK response, unless you aren't correctly authenticated. Be sure to use the IsSuccess property to evaluate the success or failure of the action.

Example

GET /databasemovement/v1/fetchstatus/project/12345/environment/5362377c-bc37-4f92-b30e-fe8c1e664cc0/operationactivity/55eb4327-9346-4c7b-82bd-fe8ef15112c6
```json
{
    "IsSuccess": true,
    "OperationActivityId": "6a90b45f-1764-4c97-b924-3f4671540237",
    "ErrorMessage": null,
    "VersionEOL": "9999-12-31T23:59:59.999999999",
    "ProjectId": 12345,
    "EnvironmentId": "5362377c-bc37-4f92-b30e-fe0c1e664cc0",
    "ActivityId": "55eb4327-9346-4c7b-82bd-fe8ef15112c6",
    "CompletionDate": null,
    "OperationStatus": "InProgress"
}
```

**OperationStatus property**

<table>
<thead>
<tr>
<th>STATUS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotStarted</td>
<td>The action hasn't yet been started.</td>
</tr>
<tr>
<td>InProgress</td>
<td>The action is in progress.</td>
</tr>
<tr>
<td>Completed</td>
<td>The action was successfully completed.</td>
</tr>
<tr>
<td>Failed</td>
<td>The action was halted.</td>
</tr>
<tr>
<td>SignedOff</td>
<td>The action was successfully completed and signed off.</td>
</tr>
<tr>
<td>Aborted</td>
<td>The action was canceled without automated cleanup.</td>
</tr>
<tr>
<td>RollbackInProgress</td>
<td>Reversal of the action is in progress.</td>
</tr>
<tr>
<td>RollbackFailed</td>
<td>Reversal of the action was halted.</td>
</tr>
<tr>
<td>RollbackCompleted</td>
<td>Reversal of the action was successfully completed.</td>
</tr>
</tbody>
</table>
This topic provides an overview of the demo data that is available.

Demo data is the base data set that is released for implementation support and demonstration purposes. The current demo data set supports the following verticals:

- Commerce
- Distribution
- Service Industries
- Public Sector
- Discrete & Process Manufacturing

The demo data set supports 40 languages across 16 countries or regions. It also supports various implementation scenarios. The following table lists the legal entities that are included in the demo data set.

<table>
<thead>
<tr>
<th>LEGAL ENTITY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRMF</td>
<td>Contoso Entertainment System Brazil</td>
</tr>
<tr>
<td>CNMF</td>
<td>Contoso Entertainment China</td>
</tr>
<tr>
<td>DAT</td>
<td>Default Company</td>
</tr>
<tr>
<td>DEMF</td>
<td>Contoso Entertainment System Germany</td>
</tr>
<tr>
<td>FRRRT</td>
<td>Contoso Retail FR</td>
</tr>
<tr>
<td>FRSI</td>
<td>Contoso Consulting FR</td>
</tr>
<tr>
<td>GBSI</td>
<td>Contoso Consulting GB</td>
</tr>
<tr>
<td>GLCO</td>
<td>Contoso Group</td>
</tr>
<tr>
<td>GLMF</td>
<td>Contoso Entertainment System</td>
</tr>
<tr>
<td>GLRT</td>
<td>Contoso Retail</td>
</tr>
<tr>
<td>GLSI</td>
<td>Contoso Consulting</td>
</tr>
<tr>
<td>INMF</td>
<td>Contoso India</td>
</tr>
<tr>
<td>ITCO</td>
<td>Contoso Italy</td>
</tr>
<tr>
<td>JPMF</td>
<td>Contoso Entertainment Japan</td>
</tr>
<tr>
<td>MXMF</td>
<td>Contoso Entertainment System Mexico</td>
</tr>
</tbody>
</table>
Embedded analytics

Demo data has been updated in five companies to provide better reports on the new embedded analytics within workspace. Filter the embedded analytics to the following legal entities for the improved report data:

<table>
<thead>
<tr>
<th>LEGAL ENTITY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEMF</td>
<td>Contoso Entertainment System Germany</td>
</tr>
<tr>
<td>GLMF</td>
<td>Contoso Entertainment</td>
</tr>
<tr>
<td>USMF</td>
<td>Contoso Entertainment System USA</td>
</tr>
<tr>
<td>USRT</td>
<td>Contoso Retail USA</td>
</tr>
<tr>
<td>USSI</td>
<td>Contoso Consulting USA</td>
</tr>
</tbody>
</table>

Reports from the Cash overview Power BI content are displayed in the Cash overview and Bank management workspaces.

To view the Cash flow forecasting reports with data, you must first run the forecast calculation process using the Calculate cash flow forecasts function from the Cash and bank management area. This needs to be completed for each company included in the forecast. You then need to refresh the LedgerCovLiquidityMeasurement aggregate measure on the Entity Store page.

For demonstration purposes, you can add cash flow forecasting demo data using the Generate data page from the Demo data module. This script will insert data into the cash flow forecasting tables to quickly populate information necessary for reports.

The credit and collections analytics can be viewed on the Manage customer credit and collections workspace. To view the analytics you need to refresh the CustCollectionsBIMeasurements aggregate measure on the Entity Store page.

The vendor payments analytics can be viewed on the Vendor payments workspace. To view the analytics, you
need to refresh the VendPaymentBIMeasure aggregate measure on the Entity Store page.

The Purchase performance analytics can be viewed on the Purchase spend analysis page from the Procurement and sourcing module. To view the analytics, you need to refresh the Purchase cube aggregate measure on the Entity Store page.

The Sales and profitability analytics can be viewed on the Sales and profitability performance page from the Sales and marketing module. To view the analytics, you need to refresh the Sales cube aggregate measure on the Entity Store page.

The production performance analytics can be viewed on the Production performance page from the Production control module. To view the analytics, you need to refresh the Production cube aggregate measure on the Entity Store page.

For demonstration purposes, you can add production performance demo data using the Generate data page from the Demo data module. This script will generate production orders and with associated feedback journals to populate the production performance reports with data.

The warehouse performance analytics can be viewed on the Warehouse performance page from the Warehouse management module. To view the analytics, you need to refresh the Warehouse aggregate measure on the Entity Store page.

For demonstration purposes, you can add warehouse performance demo data using the Generate data page from the Demo data module. This script will generate sales orders and warehouse work to populate the warehouse performance reports with data.

The demo data module is only available if you have the Demo data suite model deployed on the environment.

Vendor collaboration

In the USMF demo company, there are three purchase orders for vendor US-104 to use for demonstration purposes. You can log in as user ErinH, who is a contact person who has access to vendor collaboration for US-104.

Purchase order approval

In the USMF demo company, there are two purchase orders for INGA to approve. You can log in as user INGA to see the purchase orders awaiting approval.

Batch transfer rules for subledger journals

The batch transfer rules for subledger journal account entries have been changed to Scheduled batch to reflect a best practice. The batches are configured to run every 10 minutes. It is important to understand that accounting entries for all source documents will not be reflected in General ledger until the batch process has run. If you have requirements to see the immediate effect in General ledger, set the Transfer mode to Synchronous on the Batch transfer rules page within General ledger parameters. While Synchronous works well for product demos and environments with low transaction volumes, it can cause performance issues in larger transaction volume environments.
Cost accounting

Three Cost accounting ledgers are created in demo data. The Cost accounting ledger USP2 provides an E2E demo experience based on data from legal entity USP2. The Cost control unit consists of 2 Cost object dimensions (Cost centers and Product groups). Actual cost, Budget cost and Statistical measures are transferred for all 12 fiscal periods of year 2017. Overhead calculation has also been performed for all fiscal periods of year 2017.

Access level security is configured but not enabled. This is enabled in the Cost accounting parameters page.

After Access level security has been enabled, you can assign an employee to the role Cost object controller. You can log in as the employee and access the Cost control workspace. The employee can now see their Cost center performance and drill into details of how these were calculated.
How the packages are organized

The demo data packages are designed to be layered on top of each other, as shown in the following illustration.

However, the global information for one demo scenario might have completely different requirements than the global information for another demo scenario. For example, the dimensions for one scenario will interfere with the dimensions for another scenario. In this case, a separate global information package will be created, and only packages that are related to that global information can be layered on top of the package.

For example, there is currently a commercial system and shared package as well as a separate public sector system and shared package that can’t be used together.

**System and Shared package**

The base package, System and Shared, is the foundation for all other packages. This package creates legal entities, loads the global address book, and adds other shared information. It must be loaded first to support all the remaining packages. The package is named **100-System and Shared.zip**.
After the System and Shared package is loaded, you will see one or more of the following legal entities.

<table>
<thead>
<tr>
<th>LEGAL ENTITY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>HQUS</td>
<td>The US-based headquarters for your demo company. This company is based on the original USMF data, but it has been changed to remove the manufacturing focus in the name. It includes setup information that is intended for US companies.</td>
</tr>
<tr>
<td>HQEU</td>
<td>The non-US-based headquarters for your demo company. This company is based on the original DEMF data, but it has been changed to remove the manufacturing focus in the name. It includes setup information that is intended for non-US companies.</td>
</tr>
<tr>
<td>CONS</td>
<td>A small consolidations company.</td>
</tr>
<tr>
<td>PICH</td>
<td>A process industries company that is focused on chemicals.</td>
</tr>
<tr>
<td>PIFB</td>
<td>A process industries company that is focused on food and beverages.</td>
</tr>
</tbody>
</table>

**Financials**

The Financials data packages contain data for the general ledger, bank, accounts payable, tax, accounts receivable, fixed assets, and budgeting for a single company. The names of these data packages consist of `200-Financials` followed by the legal entity that the packages are intended for. For example, the Financials data package for the HQUS legal entity is named `200-Financials-HQUS.zip`.

At least two financial companies are required for cross-company tasks such as centralized payments. To facilitate cross-company tasks, all customers and vendors have been added to each legal entity. The CONS legal entity is required if you want to do consolidations.

The Financials data packages also have five inventory products to support the creation of invoices that can move through the accounts receivable and accounts payable processes. These items use a minimum of inventory and product functionality to support those processes. However, you no longer have to set up products when you want to demonstrate only Financials functionality. More complete products will be added when you import the Supply chain data packages.

**Expense management**

The expense management data packages contain data for expense management and aren't specific to project management. The names of these data packages consist of `225-Expense` followed by the legal entity that the packages are intended for. For example, the Expense management data package for the HQUS legal entity is named `225-Expense management HQUS.zip`.

**Project management and accounting**

The Project management and accounting data packages contain data for project accounting and expense management. The names of these data packages consist of `250-Project management and accounting` followed by the legal entity that the packages are intended for. For example, the Project management and accounting data package for the HQUS legal entity is named `250-Project management and accounting-HQUS.zip`.

**Supply chain**

The Supply chain data packages contain data for inventory management, product information, procurement and sourcing, sales and marketing, quality management, warehouse management, transportation management,
production control, process manufacturing, costing, and master planning for a single company. Because of the large number of entities, the Supply chain packages for some companies have been split into two packages. You must load both packages to complete the supply chain scenarios. However, you can load these packages as two separate projects.

These names of these data packages consist of 300-Supply chain followed by the legal entity that the packages are intended for. For example, the Supply chain data package for the PICH legal entity is named 300-Supply chain-PICH.zip. The supply chain package for the HQUS legal entity is split into packages that are named 300-Supply chain 1 of 2-HQUS.zip and 300-Supply chain 2 of 2-HQUS.zip.

Demo data package releases

The demo data packages will be released through LCS and will be specific to a release. Note that the contents of a given package are subject to change as we add more demo scenarios and tune the packages. Additional packages will also be released as we add more module areas and industry-specific scenarios.

Package names will include a release identifier. For example, for Finance and Operations 7.3, Demo data-7.3- will precede the package name that uses the previously described naming conventions. For example, the full name of the Financials package for the HQUS legal entity for Finance and Operations 7.3 will be Demo data-7.3-200-Financials-HQUS.zip.

Before you load the packages

Before you load the data packages, you must manually follow these steps.

1. If you want to sign in as a specific user, change the user’s email address to the sign-in address that you want to use. You can make this change in the User information data entity spreadsheet or, after you load data, on the Users page (System administration > Users).
2. Start the Ready to post batch scheduler. This batch job automatically posts transactions. You must start the scheduler in every legal entity where data should be processed. Follow the steps in the “The Ready to post process” section later in this topic.
3. If you aren’t using the en-us locale, you may need to alter the source data format to match the format the packages were built on. Once you’ve loaded a data entity, click into the source data format column which should be specified as value = Excel. From the next page, again select Excel. From within the source data formats page, the bottom fast-tab will be regional settings. Change the language locale to en-us if it isn’t specified as en-us. After loading packages, you can change it back to its original non en-us value.

Load the packages

The data packages must be loaded into a specific legal entity in a specific order. The number before the name of the package gives you guidance about the order that the data must be loaded in. For example, you must import 100-System and Shared.zip before you can load the Financials package for the HQUS legal entity, 200-Financials-HQUS.zip. Then, to add Supply chain data to the HQUS legal entity, you can load 300-Supply chain 1 of 2-HQUS.zip and 300-Supply chain 2 of 2-HQUS.zip.

Follow these steps to load the packages.

1. Start with an empty instance where no data is loaded.
2. Open the Data management workspace.
3. Select the Import tile to create an import job.
4. Enter a name for the job. For example, enter Import shared information.
5. Select Add file.
6. Select Upload and add, and browse to the data package that you want to import. You must start with the System and Shared data package.
7. Select the data package, and wait for the data to be loaded.
8. After the data is loaded, close the dialog box, and then select **Import**.
9. Repeat steps 5 through 8 for every additional package that you want to load. Be sure to switch to the legal entity that the data package is intended for.

**Loading package combinations**

The following packages can be loaded. When you import any package except the System and Shared package, you must be in the legal entity that is listed in the package name. The System and Shared package can be loaded from any legal entity. However, it’s typically loaded from the default company, DAT.

**Commercial Data**

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 - System and Shared</td>
<td>Load this package before you any other package.</td>
</tr>
<tr>
<td>200 - Financials - HQUS</td>
<td>You can load this package alone or together with another Financials package.</td>
</tr>
<tr>
<td>200 - Financials - HQEU</td>
<td>You can load this package alone or together with another Financials package.</td>
</tr>
<tr>
<td>200 - Financials - CONS</td>
<td>You can load this package alone or together with another Financials package.</td>
</tr>
<tr>
<td>200 - Financials - PICH</td>
<td>You can load this package alone or together with another Financials package.</td>
</tr>
<tr>
<td>200 - Financials - PIFB</td>
<td>You can load this package alone or together with another Financials package.</td>
</tr>
<tr>
<td>225 - Expense Management - HQUS</td>
<td>Follow the prerequisite step before starting the import. Load this package after the HQUS Financials package.</td>
</tr>
<tr>
<td>250 - Project management - HQUS</td>
<td>Load this package after the HQUS Financials or Expense package. If you want to also use Supply chain, you must import this package first.</td>
</tr>
<tr>
<td>300 - Supply chain 1 of 2 (base) - HQUS</td>
<td>Load this package after the HQUS Financials package. If you want to also use Project management, you must import the Project management package first before you import this package.</td>
</tr>
<tr>
<td>310 - Supply chain 2 of 2 (Discrete) - HQUS</td>
<td>Load this package after the HQUS Supply chain base package.</td>
</tr>
<tr>
<td>300 - Supply chain - PIFB</td>
<td>Load this package after the PICH Supply chain package.</td>
</tr>
<tr>
<td>300 - Supply chain - PICH</td>
<td>Load this package after the PIFB Supply chain package.</td>
</tr>
<tr>
<td>900 - Financial transactions - HQUS</td>
<td>Load this package after the HQUS Financials package.</td>
</tr>
<tr>
<td>900 - Financial transactions - HQEU</td>
<td>You can load this package alone or together with another Financials package.</td>
</tr>
</tbody>
</table>

**Public Sector data**
To load the data correctly, you need to load one package at a time, import it, and then load the next one once the import is complete. We are considering additional methods for loading the demo data to improve the process.

**Troubleshooting and known issues**

**NOTE**

We discovered an issue with the Number sequence references entity that causes a random failure during import although the data in the packages is correct. If you see an error during the import of number sequence references, follow these steps to process the failed records.

1. Click on the name of the entity (*number sequence references*) to display a form that lists all of the records in the data package.
2. Click on *Copy data to target*.
3. Change the *Run for* value to *Criteria* and *Change Rows with previous errors* to *Yes*.
4. Click *Ok* and then click *Run* on the form that appears.
5. Repeat these steps until all records import without error.

There is currently an issue that some data entities have the same name, which can cause an import failure for *document types* and *date intervals* in the **200 - Financials** packages. If you see an error during the import of these entities extract them from the .zip file provided and manually import, making sure to use the *document types* pointing to *DocuTypeEntity* and *date intervals* pointing to *LedgerDateIntervalEntity*. Once these are imported you can retry the failed records from the **200 - Financials** packages.

**After you load the packages**

Check *Ready to post* if anything needs to be posted. In some cases individual demo scripts may recommend some setup be done prior to loading this data.

In some cases, there might be data that you want to add because of a special scenario or a missing entity. Add that data after you've finished loading the data packages. You might also want to manually post additional transactions or add your own data packages to enhance the demo experience.

After you load the data packages, you must also manually follow these steps.

1. Start the workflow jobs. Select *System administration > Workflow infrastructure configuration*, and then select *OK*.
2. Set up policy precedence rules. Select *Procurement and sourcing > Setup > Policies > Purchasing policies*, and then select *Parameters*. Then select *Companies*, and move it to the right column.
3. Setup policy precedence prior to importing Expense packages. Select *Expense management > Setup > Policies > Expense report*, and then select *Parameters*. then select *Companies*, and move it to the right column.
4. After you load the Project management and accounting packages, you must run the *Resource capacity roll-up* batch job. You can run this job from the *Synchronize resource capacity roll-ups* page (*Project management and accounting > Periodic > Capacity synchronization > Synchronize resource*...
capacity roll-ups). Specify an end date that lets you schedule resources a long time in the future. After the batch job is run, automatic generation of team functionality will be enabled in the project's work breakdown structure (WBS).

5. Add Print management settings for each module.

## Transactions and automatic posting

Many scenarios for demo data require that transactions be processed after they are imported. You can process transactions by using the Ready to post feature. This feature includes both a page that lets you define the transactions that should be posted, and an entity that lets you import the definitions and automatically run them.

The following transaction types are supported when demo data is posted.

<table>
<thead>
<tr>
<th>DOCUMENT</th>
<th>ENTITY DOCUMENT ID</th>
<th>DATE FILTER</th>
<th>ID FILTERS</th>
<th>OTHER FILTERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Budget registry update</td>
<td>BudgetRegistryUpdate</td>
<td>Default date</td>
<td>Budget entry number</td>
<td>Not in use, Status = Draft</td>
</tr>
<tr>
<td>Costing version</td>
<td>CostingVersion</td>
<td>Not applicable</td>
<td>Version ID</td>
<td>Version Activation blocked = No</td>
</tr>
<tr>
<td>Customer payment journal</td>
<td>CustomerPaymentJournal</td>
<td>Transaction date</td>
<td>Journal number</td>
<td>Not posted, not workflow, not system blocked</td>
</tr>
<tr>
<td>Daily journal</td>
<td>GeneralJournal</td>
<td>Transaction date</td>
<td>Journal number</td>
<td>Not posted, not workflow, not system blocked</td>
</tr>
<tr>
<td>Fixed assets journal</td>
<td>FixedAssetsJournal</td>
<td>Transaction date</td>
<td>Journal number</td>
<td>Not posted, not workflow, not system blocked</td>
</tr>
<tr>
<td>Free text invoice</td>
<td>FreeTextInvoice</td>
<td>Invoice date</td>
<td>Not applicable</td>
<td></td>
</tr>
<tr>
<td>Inventory adjustment journal</td>
<td>InventoryAdjustmentJournal</td>
<td>Transaction date</td>
<td>Journal number</td>
<td>Not posted</td>
</tr>
<tr>
<td>Invoice journal</td>
<td>InvoiceJournal</td>
<td>Transaction date</td>
<td>Journal number</td>
<td>Not posted, not workflow, not system blocked</td>
</tr>
<tr>
<td>Price calculation</td>
<td>PriceCalculation</td>
<td>Not applicable</td>
<td>Version ID</td>
<td>Version Activation blocked = No</td>
</tr>
<tr>
<td>Purchase order</td>
<td>PurchaseOrder</td>
<td>Delivery date</td>
<td>Purchase order ID</td>
<td>Able to confirm/PR/Vendor confirm/invoice</td>
</tr>
<tr>
<td>Sales order</td>
<td>SalesOrder</td>
<td>Delivery date</td>
<td>Sales order ID</td>
<td>Able to confirm/packing slip/invoice</td>
</tr>
</tbody>
</table>
The Ready to post process

The Ready to post feature uses a batch to monitor the list of transaction types that you want to post. When the monitor detects a transaction of the type that you want to post, it uses the transaction type to generate a batch that posts those transactions. The batch that is created is the same type of batch that is created when you use the user interface for that transaction type. When the transaction batch is completed, the Ready to post monitor updates the list with the results of the processing. It also adds links to the batch and the original transaction.

Use the Ready to post page to process transactions

1. Select System administration > Periodic tasks > Batch job ready to post to open the Ready to post page.
2. Select Create posting monitor, and set up the batch parameters so that a recurring batch is running. You must complete this step only one time for each legal entity to start the batch process for posting.
3. Select New, and enter a name for the demo data job. The job name must be unique across all companies.
4. Select Add line to add a transaction type.
5. Select the transaction target. For journals, the target is Post. For other transactions, the target depends on the transaction type.
6. Specify a start date and an end date (that is, a date range) to limit the transactions that will be processed (when available).
7. Specify a “from” document and a “to” document (that is, a document range) to limit the transactions that will be processed (when available).
8. Select Add line to add additional transaction types. You can use the same type on multiple lines.
9. Select Mark ready to post. The batch status is changed from Open to Ready, and the posting monitor starts to process each line.
10. If you want to process a document immediately, select Process documents. The batch status is changed to Scheduled, and a batch is started without using the posting monitor.

When the batch is running, the status is changed to In Progress.

When the batch is completed, the status is changed to Successful or Error, depending on the results. The posting results are shown at the bottom of the page.

Use the Ready to post entity to process transactions

An entity that is named Demo data posting lets you import a list of document types that you want to post. The entity will create a demo data job on the Ready to post page. If you’ve started the posting monitor, the transactions are automatically posted after you import the data by using the entity.

The following columns appear in the Ready to post entity.
<table>
<thead>
<tr>
<th>COLUMN</th>
<th>PURPOSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DemoDataJob</td>
<td>The unique ID of the demo data job to run. Use the same ID for every line that belongs to a single job.</td>
</tr>
<tr>
<td>LineNum</td>
<td>The order that the tasks will be run in.</td>
</tr>
<tr>
<td>ProjectId</td>
<td>A link to the data project that contains the <strong>Ready to post</strong> entity. This information is for export only.</td>
</tr>
<tr>
<td>DemoDataJobStatus</td>
<td>The status of your demo data project. This information is for export only.</td>
</tr>
<tr>
<td>Document</td>
<td>The document type to process.</td>
</tr>
<tr>
<td>DocumentTarget</td>
<td>The process to run. For journals, the target must be <strong>Post</strong>. For transactions such as sales orders, the target will match the options that appear on the page when you add the task.</td>
</tr>
<tr>
<td>EndDate</td>
<td>An optional end date that limits the transactions that are processed.</td>
</tr>
<tr>
<td>FromDocument</td>
<td>An optional &quot;from&quot; document that limits the transactions that are processed.</td>
</tr>
<tr>
<td>ProcessOnImport</td>
<td>If you change the value to <strong>Yes</strong>, the demo data job will be set to <strong>Ready</strong>, and the process monitor will pick it up automatically. No action is required.</td>
</tr>
<tr>
<td>StartDate</td>
<td>An optional start date that limits the transactions that are processed.</td>
</tr>
<tr>
<td>ToDocument</td>
<td>An optional &quot;to&quot; document that limits the transactions that are processed.</td>
</tr>
</tbody>
</table>

Insert the **Ready to post** entity at the end of your data project, after all the transaction entities. In the data project, specify a sequence number that is larger than the sequence numbers that are used for the transactions entities.

If you have a mixture of transactions, some of which should be processed whereas others should not be processed, you must use date and document ranges to limit the transactions that are processed. If you can’t use the ranges, you must use a separate data package for the unposted transactions.
This topic explains how to deploy a demo environment on Microsoft Azure using Microsoft Dynamics Lifecycle Services (LCS). This topic applies to deploying a demo environment for:

- Dynamics 365 Finance
- Dynamics 365 Supply Chain Management
- Dynamics 365 Commerce

Prerequisites

Before you begin your deployment, the following prerequisites must be in place:

- Verify that you have an Azure subscription, and that you are a co-administrator on it.
- Verify that you have access to an LCS project and permissions to deploy an environment.
- Verify that you've connected your Azure subscription to your LCS project by using the information in the Complete the Azure Resource Manager (ARM) onboarding process topic.

Deploy a demo environment

Use this procedure to deploy a demo environment on Azure using LCS.

1. In LCS, open your project, and then, in the Environments section, click the plus sign (+).
2. Select the Azure environment topology, and then select Demo.
3. Select a topology.
   - For Finance and Operations, select the most recent Azure Resource Manager (ARM) topology for Finance and Operations.
   - For Commerce, select Dynamics 365 for Commerce - Demo.
4. In the Deploy environment dialog box, enter the name of the environment. This name should be unique in the Azure subscription. To make environments easy to identify, consider forming an acronym using the user’s name and the topology.
5. Select the size of the virtual machine (VM). You must use Ev3-series sizes for Finance and Operations workloads. We recommend Ev3. If you experience allocation failures, see the Azure troubleshooting guide.
6. Set the Instances field to 1.

**NOTE**

The size of the VM and the number of instances affect the cost of your subscription. For more information, see Azure pricing.

7. Click Advanced settings to add customizations to your deployment. For the demo environment, we recommend that you keep the default settings.
8. Agree to the licensing and pricing terms, and then click Next.
9. In the Confirm message box, click Deploy.

10. Open the Cloud hosted environments page to view the status of the deployment. After the deployment is successfully completed, the environment will be ready.

Log on to your demo environment

To log on to your demo environment, do the following.

1. In LCS, open the Cloud-hosted environments page, and select the demo environment that you just deployed.

2. Scroll to the right and in the Environment details pane, under Cloud services, click the appropriate link:
   - Log on to Finance and Operations
   - Log on to Commerce
Working with Microsoft to deploy Finance and Operations apps in the cloud requires that you understand the environment and subscription that you are deploying to, who can perform which tasks, and the data and customizations that you need to manage. We recommend that you sign up for the Full Microsoft FastTrack for Dynamics 365 to help speed your deployment and implementation - it’s a program that provides training and consulting to help you realize business value faster. For more information, see Microsoft FastTrack. If you choose to use the Essentials FastTrack program instead, you will be using the Implementation Project Methodology in Lifecycle Services (LCS) to help you manage your implementation project.

Customer lifecycle, subscriptions, and environment types

Microsoft assumes that all customers will follow a lifecycle similar to the following for all cloud deployments, and therefore need different environment topologies at each phase.

- Evaluate
- Develop customizations, if needed.
- Curate a “golden configuration” environment that contains only module configurations without master or transactional data. This is to be the baseline for your data migration testing and eventual go live.
- Install and test customizations and partner solutions on a tier-1 sandbox (Development or test environment).
- Test customizations, partner solutions and data configuration on a tier-2 sandbox environment.
- Deploy customizations and data configurations to a production environment with high availability.

At some phases of a project, you may have all of the environments live at once. For more information, about the default licenses and tiers that are available, see the Dynamics 365 Licensing Guide.

You may notice the terms cloud hosted or Microsoft subscriptions. A cloud hosted subscription means that the customer or partner brings their own Azure subscription and deploys Finance and Operations apps to it, for evaluation and development purposes only. The customer or partner pays for the resources deployed to their Azure subscription based on the Azure price list. A Microsoft subscription means that the customer purchases Finance and Operations licenses, which will then allow them to deploy environments to an Azure subscription which is managed by Microsoft, therefore, the customer has no separate Azure billing.

With each Enterprise offer, two environments are included by default:

- One Tier 2 sandbox (multi-box environment) for user acceptance testing (UAT).
- One production environment with high availability (HA).

Additional environments may be purchased as add-ons. For information about licensing and what is included in Microsoft Dynamics 365, see the Dynamics 365 Licensing Guide.

Here’s how the lifecycle maps to the available environments. If you already have environments deployed in your Lifecycle Services project, you can find the Environment Type and Environment Sub type on each environment’s details page.

<table>
<thead>
<tr>
<th>LIFECYCLE PHASE</th>
<th>ENVIRONMENT TIER</th>
<th>SUBSCRIPTION</th>
<th>ENVIRONMENT TYPES</th>
<th>ENVIRONMENT SUB-TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation and analysis</td>
<td>Tier 1 sandbox</td>
<td>Cloud hosted</td>
<td>Customer-managed</td>
<td>Demo</td>
</tr>
<tr>
<td>LIFECYCLE PHASE</td>
<td>ENVIRONMENT TIER</td>
<td>SUBSCRIPTION</td>
<td>ENVIRONMENT TYPES</td>
<td>ENVIRONMENT SUB-TYPE</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>------------------</td>
<td>----------------------------</td>
<td>----------------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Customize</td>
<td>Tier 1 sandbox</td>
<td>Cloud hosted or VHD</td>
<td>Customer-managed</td>
<td>Develop</td>
</tr>
<tr>
<td>Golden configuration</td>
<td>Tier 1 sandbox</td>
<td>Cloud hosted</td>
<td>Customer-managed</td>
<td>Develop</td>
</tr>
<tr>
<td>User acceptance testing (UAT)</td>
<td>Tiers 2-5 sandbox</td>
<td>Microsoft</td>
<td>Microsoft-managed or self-service</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Go live</td>
<td>Production</td>
<td>Microsoft</td>
<td>Microsoft-managed or self-service</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

Tiers 2-5 can be purchased to increase performance of the environment. The higher the tier, the more compute and database capacity is reserved for your use. For more information about Self-service environment types, see Self-service deployment overview.

**IMPORTANT**

Tier 1 sandbox environments are no longer Microsoft-managed starting in November 2020. For demo, build, and develop purposes the Tier 1 environments can be deployed on a customer's Azure subscription directly from Lifecycle Services (LCS).

Environment lifecycle operations

Users with the Environment Administrator or Project Owner roles in Lifecycle Services can perform various lifecycle operations on their environments. These operations often involve downtime on the environment until the task is finished. Each of these operations are located under or next to the **Maintain** button on each environment details page.

<table>
<thead>
<tr>
<th>LIFECYCLE OPERATION</th>
<th>DESCRIPTION</th>
<th>LEARN MORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply software</td>
<td>Install Microsoft updates, ISV solutions, or your own customization packages.</td>
<td>Apply updates to cloud environments</td>
</tr>
<tr>
<td>Enable access</td>
<td>Allow list your IP for Remote Desktop or database access</td>
<td>See the Remote Desktop section later in this topic</td>
</tr>
<tr>
<td>Restart services</td>
<td>Ability to restart components of your environment</td>
<td>Restart environment services</td>
</tr>
<tr>
<td>Move database</td>
<td>Full data lifecycle management</td>
<td>Database movement operations</td>
</tr>
<tr>
<td>Maintenance mode</td>
<td>Ability to change configuration with only admin access</td>
<td>Maintenance mode</td>
</tr>
<tr>
<td>Upgrade</td>
<td>Upgrade code and data from 7.x to the latest version</td>
<td>Process for moving to the latest update</td>
</tr>
<tr>
<td>Deallocate</td>
<td>Ability to turn off an environment not being used, or to troubleshoot a failed action</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>
LIFECYCLE OPERATION | DESCRIPTION | LEARN MORE
--- | --- | ---
Start | Ability to turn on an environment for use | Not applicable
Delete | Ability to delete an environment previously deallocated | Not applicable

Security and compliance

Finance and Operations is PA-DSS 3.1 certified which means that all communications between components are secured out-of-the-box.

All Finance and Operations front-end virtual machines in Microsoft Azure are configured during deployment to only accept TLS 1.2.

IMPORTANT
Customers who have administrator access to Microsoft-managed sandboxes, including any add-on sandboxes purchased, must follow these guidelines:

- By default, automatic Windows update is enabled for all Tier 1 - 5 sandboxes and should NOT be disabled. This ensures that any time that Microsoft pushes security or critical infrastructure updates to your environment, your environment receives the latest set of updates and is updated each month with the operating system fixes that Microsoft releases.
- Admin passwords on these environments should NOT be changed. Environments that have admin passwords changed will be flagged by Microsoft. Microsoft reserves the right to reset the admin password, and will do so.
- Adding new user accounts to any Microsoft-managed VM is NOT permitted. Microsoft reserves the right to do this, and will remove the newly added user accounts without providing notice.

Finance and Operations is not covered by a FedRAMP ATO at this time. If Finance and Operations is provisioned in the United States, all customer data at rest is stored in data centers located in the United States, as described in International availability of Dynamics 365.

Remote Desktop

Microsoft-managed environments

WARNING
Microsoft will be removing the use of Remote Desktop by customers and partners. Each environment will first have administrator access removed, but still allow non-administrator access to the virtual machines. After this, all access will be removed. For each step of this phased removal, an email notification will be sent to the Notification list setup for each environment. All Remote Desktop access will be removed by November 2020.

Customers are required to complete additional setup to connect to virtual machines (VMs) through Microsoft Remote Desktop (RDP). This additional setup applies to all Microsoft-managed environments, including Tier 1 through Tier 5 sandboxes and add-ons. In order to connect to Tier 1 through Tier 5 sandbox environments, you must explicitly enable access (safe list) from your organization's IP address space. This can be done by a Lifecycle Services (LCS) user who has access to the Environment page (Maintain > Enable Access) where they can enter the IP address space that will be used to connect to the virtual machines through Remote Desktop. Access rules are either a single IP address (example: 10.10.10.10) or an IP address range (example: 192.168.1.0/24). You may add multiple entries at once as a semicolon (;) separated list (example: 10.10.10.10; 192.168.1.0/24).
Customer-managed/Tier-1 environments

By default, Remote Desktop is enabled for all environments that are not managed by Microsoft. We recommend that customers restrict access to any environments that belong to their subscriptions. This can be done by configuring Network Security Group rules on the environments directly in Azure Portal.

Windows Remoting (WinRM)

Windows Remoting (WinRM) is disabled on all environments. Although you can enable WinRM on environments that belong to your subscriptions through Azure Portal, we strongly recommend that you do not do this.

WARNING

Exceptions to enable WinRM will not be granted for any Microsoft-managed environments.

Availability

The guaranteed uptime for Finance and Operations apps is 99.9%. Planned downtime occurs once a month and lasts no longer than eight hours. Because the work completed during the downtime doesn't always take eight hours, we will always communicate the estimated amount of time that your environments will be down. For more information, see [Get support for Finance and Operations apps or Lifecycle Services (LCS)](https://docs.microsoft.com/en-us/dynamics365/finance和支持/finance-and-operations-help).

High-availability features

To ensure service availability, all production environments are protected by using default Azure high availability (HA) features. HA functionality provides ways to avoid downtime caused by the failure of a single node within a datacenter, and DR features protect against outages broadly impacting an entire datacenter. Azure availability sets are used to prevent single-point-of-failure events. For more information about Azure availability sets, see [Use availability zones to protect from datacenter level failures](https://docs.microsoft.com/en-us/azure/virtual-machines/windows/availability-zones). High availability for databases is supported through Azure SQL. For more information, see [Overview of business continuity with Azure SQL Database](https://docs.microsoft.com/en-us/azure/dynamics365/financi管理/financi和operations/绍/financi和operations-business-continuity).

Database backup retention

Databases for Microsoft-managed or Self-service Tiers 2-5 environments have automated backups taken by Azure SQL every few minutes. These backups can be restored by using Lifecycle Services point-in-time restore.
Disaster recovery features

Production environments are configured with Azure disaster recovery support that includes the following:

- Azure SQL active-geo replication is configured for the Finance and Operations database of the production environment. For more information about SQL replication, see Compare geo-replication with failover groups.
- Geo-redundant copies of Azure blob storage (containing document attachments) in other Azure regions. For more information, see Azure Storage redundancy.
- Same secondary region for the Azure SQL and Azure blog storage replication.

Only primary data stores are supported by replication. The Financial reporting services and Entity store database use transformed data from the primary database and must be generated after the recovery site has been set up and the Finance and Operations service has started.

Service availability in Azure regions

Finance and Operations apps can be deployed into a subset of Microsoft Azure datacenters using Dynamics Lifecycle Services (LCS). Azure is generally available in datacenters and geographical locations around the world. With Finance and Operations apps, customers can specify the region or datacenter where their customer data will be stored. Microsoft may replicate data to other regions for data durability, but we will not replicate or move customer data outside the geographical location. For more details, see Service description for Finance and Operations apps.

IMPORTANT

Regardless of where customer data is stored, Microsoft does not control or limit the locations from which customers or their end-users may access it. For more information, see International availability of Dynamics 365.

Upcoming changes to region availability

Dynamics 365 solutions consist of a collection of multiple services. Looking across Dynamics 365 applications, the Power Platform and the Azure services that they both depend on, the required matrix of services is quite large and growing. We have locked on a strategy of selecting a subset of data center regions across the globe to simplify ensuring that we have availability of the full portfolio of required services. Our plan is to optimize to have minimal latency between the component services of a solution and as a result, we are focused on having the full portfolio of services available in each of the designated data centers.

Additionally, the Finance and Operations architecture is being enhanced to build on self-service for greater elasticity, stronger reliability, and more seamless maintenance. Customers gain material efficiency by having deeper self-service deployments in fewer data centers. This transition also benefits from selecting a subset of Azure regions. To that effect, the regional availability of Finance and Operations apps will now be limited to East US, West US, and Central US in North America for all new projects. For a list of the latest supported regions, see International availability of Dynamics 365.

Support for East US2, West US2, West Central US, North Central US, and South Central US will continue to be available for projects and environments that currently have their data stored in those regions on Microsoft-managed environments.
Frequently asked questions

Why does the status display 'Maintenance' on my environment in LCS?
To provide the best experience and performance, Microsoft performs maintenance operations on your environment. During some of these maintenance operations, your environment status may display one of the following statuses:

- Preparing for maintenance
- Prepared for maintenance
- Maintenance in progress

While your environment is in this state and until the status returns to 'Deployed', you will not be able to perform any lifecycle operations, such as package applications. There will be no impact to Finance and Operations apps. Users can continue with normal operations without any service interruption. You will receive an email notification before any maintenance operation puts your environment in this state.

How do I connect to the SQL database on my sandbox environment?
To connect to the SQL database in your sandbox environment, follow the steps in Enable just-in-time access.

How do I access a development instance?
For information about how to access development instances, configure on-premises development VMs, and find configurations settings for developers and administrators, see Deploy and access development environments.

How do I deploy a demo environment?
A demo environment includes only Microsoft demo data. You can use a demo environment to explore default features and functionality. For more information, see Deploy a demo environment.

How do I move my customizations between environments?
To move customizations from a development to a sandbox or production environment, see Create deployable packages of models.

Can I bring my own domain name?
You can bring your own domain name if it is running Azure Active Directory (Azure AD), and the administrator of your Azure AD instance has enabled the Finance and Operations apps within their Azure AD. This is usually done through the office email, after you buy a license. When you click the link to accept the offer, Azure AD is set up for you.

Can I add guest Azure AD accounts as users?
You can add guest Azure AD accounts if you have correctly configured them within Azure AD and enabled the Finance and Operations apps within your Azure AD.

Why am I no longer able to see the Private AOS machines in one or more of my Tier 2 through Tier 5 sandbox environments?

If there are other customer workloads that are not part of the Dynamics 365 or Power Platform family that also require proximity to the Dynamics 365 and Power Platform services, Microsoft will work with customers to coordinate a plan for the overall migration. For more information, see Cloud deployment overview: Frequently asked questions.
The Private AOS VMs were part of your environment configuration as they were needed to secure communication between the AOS and BI machines in the past. With recent updates, all communication between AOS and BI machines are secure directly and no longer need the intermediary Private AOS machines. Therefore, we are in the process of rolling out removing the Private AOS machines. As we are removing the machines in batches, you may notice that only some of your environments have the Private AOS machines removed. This change will not impact functionality or security in any way and will be transparent to you.

Why am I no longer able to Remote Desktop into one or more of my Tier 1 through Tier 5 Microsoft-managed sandbox environments?

Microsoft-managed Tier 1 through Tier 5 sandbox environments require Remote Desktop management endpoints to be restricted to specific IP Address sets (safe list). Microsoft regularly validates that the environments are sufficiently restricted. Microsoft reserves the right to immediately remove any IP Address safe list rules that violate the above guidelines without notice. You may not be able to Remote Desktop into your environment for one of these reasons:

- Your current IP address is not in the safe list.
- Your IP has changed from the IP address listed in the safe list.
- Microsoft deleted the rule containing your IP address from the safe list because it violated a guideline.

To regain access to the environment, you will need to add the IP address of the computer from which you are connecting to. To do this, complete the steps Remote Desktop section earlier in this topic.

When will the availability of reduced regions go into effect for new onboarding?

Beginning August 1, 2020, new projects for Finance and Operations will be on boarded to the following regions:

- East US
- West US
- Central US

**IMPORTANT**

Central US is no longer an option for Self Service migrations beginning April 1, 2021.

- East US2
- West US2
- West Central US
- North Central US
- South Central US

This will not affect any environments that have their data stored in the deprecated regions before August 2020. In the near future there is a transition plan to move customers in the deprecated regions into the reduced regions.

I’m unable to redeploy an environment after deleting it, the environment slot is missing.

This is due to the license expiring, which means that you no longer have the minimum required licenses to obtain an environment slot. Please review your subscription status and then reactivate the expired license to enable the redeployment.
Self-service deployment overview

IMPORTANT
Functionality noted in this topic will be made available to users based on the geographic location recognized by Microsoft Azure.

Self-service deployment is available for cloud environments. Self-service deployment enables easier deployment and significantly reduced deployment times.

IMPORTANT
The functionality for this feature will be released incrementally, based on your Microsoft Azure country/region. However, this functionality is currently available only for new customers who are in the process of signing up for Finance and Operations apps. There is no change in existing environments for current customers.

Note that not all new customers will see this functionality. However, the number of new customers who have access to it will gradually increase.

What’s new or changed

Customers using the self-service capabilities will see the following changes in their Lifecycle Services (LCS) experience.

- Deployment is self-service and can be completed within an average time of 1-2 hours, depending on the type of environment (sandbox or production). There are no longer lead times and wait times for deployment. You can control when you deploy, and verify that the environment is deployed. This experience is the same as the current experience. For more information, see Self-service deployment FAQ.

- You will no longer have remote desktop access to the Tier 2+ sandbox environments. All operations that need remote desktop access are now available as self-service actions. The following image shows some of the operations in the environment’s Maintain > Move database menu option. For more information, see Maintenance operations for deployments.
Remote desktop access will be restricted only to environments deployed using the self-service deployment. There is no change to existing environments or existing customers.

- The diagnostics capabilities will remain the same, which enables troubleshooting without remote desktop access. For more information, see Troubleshoot environments deployed through self-service deployment.

- You will not have SQL Server access on Tier 2+. You will continue to have SQL database access using just-in-time access.

- You will need to provide a combined deployable package for customizations. That is, all custom extension packages, including ISV packages, must be deployed as a single software deployable package. You will not be able to deploy one module at a time. This was always a recommended best practice and is now enforced.

- The document preview experience has been improved to deliver greater fidelity with the printed output. Before this change, documents viewed on screen were displayed using an HTML viewer. Although the HTML format supported interactive functions like embedded drill-thru links and collapsible sections, this was not a true representation of the document rendered by the service. With the new embedded PDF Viewer, customers have access to a preview that is consistent with the printed documents. For more information, see Preview PDF documents with an embedded viewer.

- Custom fonts are no longer supported for document reports rendered using the built-in SSRS framework. Finance and Operations apps include access to hundreds of standard, business-ready fonts available for documents rendered by the cloud-hosted service. This portfolio will continue to grow as the service expands into new regions and industries. However, the service no longer supports the installation of custom fonts in customer environments. Requests to expand the collection of fonts supported by the
service will be considered on a case-by-case basis.

- The service no longer supports business logic defined using Visual Basic script embedded in SQL Server Reporting Services (SSRS) reports. Visual Basic expressions defined in Tablix controls used to format and evaluate data at runtime will continue to be fully supported. However, the service will ignore instructions defined in Visual Basic script functions. This change was necessary to improve the security and reliability of the service.

- Sub reports are no longer supported in document reports defined using the SSRS development tools. Application solutions that include sub reports will need to be recreated or replaced with solutions that take advantage of other reporting options supported by the service.

**IMPORTANT**

Support for sub report items has been re-introduced with the Platform update 36 release. Customers dependent on solutions that include properly formatted sub report items can transition to self-service deployments running on Platform update 36 or later.
This topic walks through the process of deploying sandbox (Tier 2 and above) and production environments with the self-service deployment experience. Refer to the following procedure to deploy these environments.

1. Select **Configure** on the project dashboard page.
2. Select the **Application** and **Platform** version for the environment that you want to deploy.
3. Provide a unique name for the environment.
4. Select the **region** where you want this environment to be deployed.
5. Choose whether you want to load **demo data** in your environment or if you want an empty database.
6. Select the **BPM library** that will be set as the Getting started library in the product.
7. Select from a list of available **AOT packages** (customization packages) on the Software Deployable tabs in the Asset Library if you want to apply customizations. Only packages generated from a build environment on version 8.1 and above should be selected. Applying a package from an incompatible version will have an adverse effect on the environment.
8. Specify **two user email addresses** that will receive notifications related to this environment. These users are in addition to the users who are already on the project team (such as an ISV or a partner).
9. Select the **email address** of the **user** that will be set as the **system administrator** in the product.
10. After you validate the configurations, click **Submit** to trigger the deployment.
11. If you plan to use channels, you must also **Initialize Retail Cloud Scale Unit**.

The environment deployment starts immediately and could take anywhere between 30 minutes to 1 hour to complete for a sandbox environment and 1-2 hours for a production environment.

To closely monitor the deployment progress, you can view the **Environment details** page. The environment state should change to either **Deploying** or **Deployed/Failed**.

If the deployment succeeds, the environment state will be **Deployed**, and the user set as the environment administrator will be able to sign in to the environment.

If the deployment fails, then create a Microsoft support ticket and provide the **Last Activity ID** (available under **Manage environment** on the **Environment details** page) in the support ticket.

### Delete an environment

You can delete an environment that is in the deployed state directly through the environment details page. To delete an environment, go to the environment details page and click the **Delete** button on the action bar. A confirmation dialog box will display asking you to enter the name of the environment that you want to delete. After you enter the environment name and select **Yes**, the deletion operation will start. When the deletion operation is complete, the **Configure** button for this environment will be enabled on the project dashboard if you want to redeploy the environment.
**NOTE**
To redeploy an environment, you will need to delete the environment and then deploy it again using the steps listed above.

**IMPORTANT**
For existing customers using channel functionality in the cloud, to ensure continued and uninterrupted support for your business, we require that you *Initialize Retail Cloud Scale Unit* no later than January 31, 2020. There is no action required for customers who exclusively use Store Scale Unit. Contact your Microsoft FastTrack Solution Architect if you require an extension.
**Self-service deployment FAQ**

11/24/2021 • 8 minutes to read • Edit Online

---

**IMPORTANT**

Functionality noted in this topic will be made available to users based on the geographic location recognized by Microsoft Azure.

---

This topic provides answers to some frequently asked questions about self-service deployment. Refer to the known issues list if your scenario is not listed here.

Why do I see only application version 8.1.1 and Platform update 21 and above when I try to deploy my sandbox environment using self-service deployment?

Currently, self-service deployment supports only application version 8.1.1 with Platform update 21 and above.

My development environment is on application version 8.1. Am I still able to move my customization to the sandbox environment?

Yes. Application version 8.1.1 is fully backward compatible with application 8.1.

What is the minimum supported application and platform version when trying to use the self-service deployment?

Application version 8.1 with Platform update 20 is the minimum supported version.

What do I do if deployment fails?

1. Delete the deployment.
2. If you want to reuse the same environment name, wait 5 minutes and try again. Otherwise, retry deploying with a new name.
3. If deployment fails again, log a Support ticket.

**NOTE**

Microsoft will add automatic retry logic in an upcoming release and make logs available.

What if my deployment fails with an “environment already exists” error?

You may be trying to reuse the environment name of a deployment that you just deleted. Wait 5–10 minutes before retrying the deployment.

I don't have Remote Desktop access to my sandbox environment. How do I perform critical actions that require Remote Desktop access?
today?

Although you will no longer have Microsoft Remote Desktop access, you can continue to operate your Tier 2+ sandbox environments, because Microsoft is providing self-service capabilities and tools that you can use to perform the common critical actions that are described here.

**Access the Azure SQL database**

You can access the Microsoft Azure SQL database by following these steps.

1. From LCS, add a safe list of the IP address of the machine that you will use to connect to the Azure SQL database using SQL Management Studio.
2. Use LCS to request access to see the database credentials. You must provide a reason for requesting access.

As soon as you submit the request, it’s automatically approved. Within a minute or two, you will be able to see the database access credentials on the LCS environment details page. You can use the credentials to connect to the SQL database.

**NOTE**

The credentials are valid for eight hours, and then they will expire. After the credentials expire, you will have to request access again.

**Access log files**

You can view and download all log files from the Activity tab on the LCS environment monitoring page.

**Use perfmon tools**

Although you can no longer use Remote Desktop and then use perfmon.exe to diagnose performance issues, you can monitor critical health metrics for CPU and memory consumption through LCS. In addition, you can run predefined queries on demand, and you can run predefined actions to mitigate performance issues on your Tier 2+ environments.

**Access self-service logs**

All logs that are related to self-service operations that are performed on the environment through LCS are available for download from LCS. These self-service operations include deployment, servicing, and database movement. You can download the log folders from the LCS environment history page.

**Turn Maintenance mode on/off**

Starting in the January 2019 release, you will be able to turn Maintenance mode in your environment on and off through a self-service action in LCS.

**Restart services**

Starting in the January 2019 release, you will able to restart services through a self-service action in LCS.

**Configure the Regression suite automation tool**

Microsoft is working on tooling that will let you update certificate thumbprints in the wifconfig file without having to use Remote Desktop. This tooling is scheduled for release in February 2019. If you must use the Regression suite automation tool before then, log a support request.

I must perform one of the critical actions that are listed earlier in this topic, but the self-service feature isn’t yet available. How do I get help?

Log a support ticket, and Microsoft will help you perform the action on your environment.
I don't have Remote Desktop access to my sandbox environment, and the critical action that I must perform isn't listed in this topic. How do I get help?

If your critical action isn't listed earlier in this topic, add a comment to this topic or log a documentation bug, and Microsoft will address your requirement.

What regions are supported on self-service in North America?

We currently only support the following regions in North America.

- East US
- West US
- Central US

**NOTE**

Central US is no longer being provided as an option for self-service migrations beginning April 1, 2021.

For more information about region availability, see International availability of Dynamics 365.

My environments are currently in the regions that are no longer supported. How will this change affect me?

Projects that have been onboarded on or after August 1, 2020 are no longer supported in the following regions:

- East US2
- West US2
- West Central US
- North Central US
- South Central US

**NOTE**

This will not affect any environments that have their data stored in the deprecated regions before August 2020. There is a transition plan to move customers in the deprecated regions into other regions. For a list of the latest supported regions, see International availability of Dynamics 365.

- With all self-service migrations, we are changing the outbound IP addresses in regions where your environments are hosted. New outbound IP addresses are available, so you must add them before your upcoming self-service migrations. For more information about IP addresses, see For my Microsoft-managed environments, I have external components that have dependencies on an explicit outbound IP safe list. How can I ensure my service is not impacted after the move to self-service deployment?.
- If you have any integrations or other dependencies that are latency-driven and have questions regarding how the change in regions will impact that, please contact Microsoft Support.
- Central US is no longer an option for self-service migrations. If a customer plans to leverage dual-write functionality, virtual entities, or any Finance and Operations apps add-ins that have dependencies on Dataverse, keep in mind that Dataverse is not supported in Central US. There are currently no plans to support Dataverse in Central US in the foreseeable future. For continued functionality of features in a supported region, we will plan to move your environment to East US or West US instead of Central US.
- If you have a project which has a few environments using self-service capabilities in Central US and others in North Central, South Central or West Central, the rest of the IaaS environments can still be moved to East or West US as part of migrations. For the environments already on Service Fabric and in Central US, please contact Microsoft Support to get them moved to East/West US based on your preference.
For my Microsoft-managed environments, I have external components that have dependencies on an explicit outbound IP safe list. How can I ensure my service is not impacted after the move to self-service deployment?

With self-service migrations, we are changing the outbound IP addresses in regions where your environments are hosted. New outbound IP addresses are available so you can add them in preparation for the upcoming self-service migrations or post migrations.

- If none of your external components have dependencies on an explicit inclusion list of IPs or special handling of outbound IP addresses for routing or firewall, no action is required.
- If any of your external components have special handling for the outbound IP addresses to communicate to the AOS, add the new outbound IP addresses where the existing ones appear. Don’t replace the existing IP addresses. You can find the new outbound IP addresses in the following list. For example, an outbound IP address may be explicitly included in a firewall outside your AOS, or an external service may have an allowed list that contains the outbound IP address for your AOS.

The inbound IP address to the AOS is dynamic. This can, and will, change over time as infrastructure changes occur.

<table>
<thead>
<tr>
<th>GEOGRAPHY</th>
<th>AZURE REGION</th>
<th>IP PREFIXES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asia Pacific</td>
<td>East Asia</td>
<td>52.229.231.64/26</td>
</tr>
<tr>
<td>Asia Pacific</td>
<td>Southeast Asia</td>
<td>20.44.247.0/26</td>
</tr>
<tr>
<td>Australia</td>
<td>Australia East</td>
<td>20.40.190.0/26</td>
</tr>
<tr>
<td>Australia</td>
<td>Australia Southeast</td>
<td>20.40.165.192/26</td>
</tr>
<tr>
<td>Brazil</td>
<td>Brazil South</td>
<td>191.234.130.0/26</td>
</tr>
<tr>
<td>Canada</td>
<td>Canada Central</td>
<td>20.151.60.0/26</td>
</tr>
<tr>
<td>Canada</td>
<td>Canada East</td>
<td>52.155.27.128/26</td>
</tr>
<tr>
<td>China</td>
<td>China East 2</td>
<td>52.131.245.128/26</td>
</tr>
<tr>
<td>China</td>
<td>China North 2</td>
<td>52.130.157.64/26</td>
</tr>
<tr>
<td>Europe</td>
<td>North Europe</td>
<td>52.155.160.192/26</td>
</tr>
</tbody>
</table>

NOTE
The outbound IP address from the AOS will be an IP address from the listed ranges based on the Azure region of your deployment. The specific outbound IP address may vary across outbound requests, even from within the same session.
What does the downtime look like for self-service migrations?

Self-service migration for any environment takes three hours of 100% downtime, with a six-hour pre-migration window leading up to the actual migration downtime of 3 hours. The environment will be available with limited servicing capabilities during the six-hour pre-migration window, but will be completely unavailable in the three-hour migration window. We recommend that customers do not schedule any servicing activity, like package deployment, during the pre-migration window because it will interfere with migrations and will trigger a migration cancellation.

Is there a potential impact on the environment's certificates?

---

<table>
<thead>
<tr>
<th>GEOGRAPHY</th>
<th>AZURE REGION</th>
<th>IP PREFIXES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Europe</td>
<td>West Europe</td>
<td>20.61.88.128/26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>51.105.159.192/26</td>
</tr>
<tr>
<td>France</td>
<td>France Central</td>
<td>51.138.205.48/28</td>
</tr>
<tr>
<td>France</td>
<td>France South²</td>
<td>52.136.140.96/28</td>
</tr>
<tr>
<td>India</td>
<td>India Central</td>
<td>20.193.248.192/26</td>
</tr>
<tr>
<td>India</td>
<td>India South</td>
<td>20.40.5.0/26</td>
</tr>
<tr>
<td>Japan</td>
<td>Japan East</td>
<td>20.48.77.192/26</td>
</tr>
<tr>
<td>Japan</td>
<td>Japan West</td>
<td>20.39.179.192/26</td>
</tr>
<tr>
<td>South Africa</td>
<td>South Africa North</td>
<td>102.133.204.16/28</td>
</tr>
<tr>
<td>South Africa</td>
<td>South Africa West²</td>
<td>102.133.78.96/28</td>
</tr>
<tr>
<td>Switzerland</td>
<td>Switzerland North</td>
<td>51.103.164.128/28</td>
</tr>
<tr>
<td>Switzerland</td>
<td>Switzerland West²</td>
<td>51.107.230.128/28</td>
</tr>
<tr>
<td>United Arab Emirates</td>
<td>UAE North</td>
<td>20.203.41.96/28</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>UK South</td>
<td>51.11.26.192/26</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>UK West</td>
<td>51.137.139.0/26</td>
</tr>
<tr>
<td>United States</td>
<td>Central US</td>
<td>13.86.98.128/26</td>
</tr>
<tr>
<td>United States</td>
<td>East US</td>
<td>52.255.218.64/26</td>
</tr>
<tr>
<td>United States</td>
<td>West US</td>
<td>52.250.195.128/26</td>
</tr>
</tbody>
</table>

¹ For Azure regions with multiple IP prefixes, such as West Europe, outbound requests will utilize IP addresses from any of the listed IP prefixes.

² Denotes a BCDR-only Azure region. Outbound requests will only originate from this region in disaster recovery scenarios that require regional failover within the geography.
Yes, if you are migrating from the previous non self-service deployment, your environment's certificate may be renewed due to infrastructure differences. Determine if there is any dependence on the certificates in your solution/integration and perform the needed actions after the migration.
This topic describes the known issues with self-service deployment.

Lifecycle Services (LCS)

Features not intended to be implemented
The following LCS features will not be implemented in self-service deployment.

- **System diagnostics** - All data and functionality provided by system diagnostics today will be available through other features in the product and LCS.
- **Service requests** - Service requests are being replaced with self-service actions.

Known issues in this release
Know issues are bugs that will be addressed in upcoming releases. Every 2 weeks there is a new release of LCS.

Finance and Operations apps

NOTE
Dynamics 365 Commerce is implemented in the modern deployment experience with the 10.0.10 release. For more information, see Create payment packaging for Application Explorer for self-service deployment.

Features not intended to be implemented
The following feature will not be implemented in self-service deployment.

- **Custom fonts** - Custom fonts are not supported. For more information, see Document Reporting Service in Dynamics 365 applications.
- **Customizations related to user interface (UI) components on self service** - Customizations that do not use the standard Financial Reporting or SQL Server Reporting Services (SSRS) in Finance and Operations apps often take a dependency on UI components of the operating system where the AOS runs. Example dependencies include Windows fonts, web browsers such as Internet Explorer, or custom PDF rendering. We do not ensure the host operating system will include any support for font infrastructure, web browsers, or any general UI components. The host operating system will change when migrating to self-service infrastructure. If you have such dependencies and have additional questions, please contact Microsoft Support.

Features no longer supported
The following feature is no longer supported with self-service deployment.

FTP
Customizations relying on FTP are not supported with self-service deployment. You should consider the following information:
We do not ensure that all outbound requests from an Application Object Server (AOS) are on a static IP address.

Until June 2021, we will ensure that all outbound requests during a particular AOS session will be on the same IP address. This can have implications for some processes, such as FTP. We recommend removing the use of FTP by using Power Apps to pull the files in and make API calls into Finance and Operations apps to import the files using the Data Integration framework. For more information, see Data entities integration overview. Some specific examples include:

- Use the native SFTP connector (as described in Monitor, create, and manage SFTP files in Azure Logic Apps), which still requires some port opening on the firewall to call the on-premises service. Consider that for Logic Apps, the list of IPs is much shorter than the entire region allowlist and the limits and configuration in Power Automate.

- Use the “Local Filesystem” connector, as described in Outbound IP addresses, in combination with the on-premises data gateway. For more information, see Install on-premises data gateway for Azure Logic Apps. This solution completely removes the need for the IP allowlist, which is deprecated in self-service deployment, while keeping a very high-level of security.

- Self-service capabilities: If FTP scenarios are failing after migrating to self-service capabilities, review the configuration at the FTP server. The most common scenario is the need to update the allowed IP list with the ranges for self-service environments.

NOTE
For more information about deprecated features, see Removed or deprecated platform features and Removed or deprecated features from previous releases.
This topic explains how to perform maintenance operations for an environment that was deployed by using the self-service deployment experience.

**Restart services**

You can use the restart services functionality to restart individual services that are associated with a Tier 2, Tier 3, Tier 4, or Tier 5 standard acceptance test (sandbox) environment that is deployed in a Microsoft subscription. The services that you can restart are AOS service, DIXF (Data import export framework service), and MR (Management Reporter service).

To restart a service, follow these steps.

1. In Microsoft Dynamics Lifecycle Services (LCS), on the environment details page, select **Maintain > Restart service**.
2. Select the service to restart, and then select **Confirm**.
   
   During the restart, the environment's status is updated to **Restarting service**, and you can't start any other maintenance operations. After the service has been restarted, the environment's status is returned to **Deployed**.

**Maintenance mode**

Finance and Operations apps includes a system-wide setting that is named maintenance mode. Maintenance mode gives system admins a safe way to make system changes that might affect system functionality. For example, configuration keys can be turned on or off. While maintenance mode is on, only the system admin and users who are assigned to the **Maintenance mode** user role can sign in to the system. By default, maintenance mode is turned off.

To turn maintenance mode on or off, follow these steps.

1. In LCS, on the environment details page, select **Maintain > Enable Maintenance mode**.
   
   The environment’s status is changed from **Deployed** to **Servicing**. After maintenance mode has been turned on, the environment's status is updated to **In Maintenance**, and only the system admin can sign in.
2. After the system admin has finished making configuration changes, on the environment details page, select **Maintain > Disable Maintenance mode**.
   
   The environment’s status is changed from **In Maintenance** to **Servicing**. After maintenance mode has been turned off, the environment's status is returned to **Deployed**. The environment history is updated to reflect the fact that the environment was put into maintenance mode.
Access database

Remote access is turned off for environments that were deployed by using the self-service deployment experience. During implementation, if you must connect to the database on your Tier 2, Tier 3, Tier 4 or Tier 5 standard acceptance test environments for troubleshooting purposes, access will be granted as it’s required. The access won’t be persistent.

To connect to a database, follow the instructions in Enable just-in-time access.
This topic walks through the process of applying updates to an environment that was deployed by using the self-service deployment experience.

**IMPORTANT**

Functionality noted in this topic will be made available to users based on the geographic location recognized by Microsoft Azure.

**IMPORTANT**

In the next-generation infrastructure, updates are applied differently than they are applied in the current flow. Whatever is provided in the package is applied to the environment, and it overwrites whatever is already present in that environment. Therefore, you must create a single deployable package that contains all customizations and independent software vendor (ISV) solutions from your build environment. If the list of models in the environment differs from the list of models in in the package, you receive a warning before the update is applied. For information about how to create a single package, see Manage third-party models and runtime packages by using source control.

Applying updates to self-service environments

Self-service environments use a special approach to performing updates, because the container-based image process is used to build the environment’s runtime. When these images are applied to a sandbox environment, the customer gives an Update name value to them, and they are shown in the environment history. An update image consists of three parts:

- **Microsoft binaries** that Microsoft releases on a regular basis, and that include new platform and application software updates. These binaries are available from the environment details page for your environment in Microsoft Dynamics Lifecycle Services (LCS). You will see a single tile that shows a cumulative binary update of all the application and platform fixes. To apply this update, select the package, and then select Save package to save the Microsoft update to the project asset library.
- **An AOT deployable package**, which is an all-in-one package that is the sum of all the custom code that the customer wants to apply to its environment.
- **An Update name value** that the customer provides in LCS.
The combination of these binaries is the basis for an image that is used to create an instance of an Application Object Server (AOS). The **Update name** value lets customers provide a meaningful name that indicates what the update contains.

**Update by using packages on Tier 1 sandbox/development and test/demo/build environments**

To apply updates to a Tier 1 sandbox/development and test/demo/build environments that were deployed through LCS, follow the steps in Apply updates to cloud environments.

**Update by using packages on Tier 2 and above Standard Acceptance Test/sandbox environments**

**IMPORTANT**

Package application causes system downtime. All relevant services will be stopped, and you won’t be able to use your environments while the package is being applied.

At the sandbox tier, updates are still applied via LCS by using the same packages that were used for the Microsoft-managed environments before self-service. When a customer applies a package, it can be either a Microsoft binary or an AOT deployable package. In both cases, Microsoft takes the latest image of the environment, overwrites the binary or AOT component, and produces a new image that is used to re-create the runtime for the environment.

Before you begin, verify that the deployable package has been uploaded to the project asset library in LCS, and that validations have succeeded.

After the package is in the project asset library, follow these steps to update your environment.

1. Open the environment details page for the environment where you want to apply the package.
2. Select Maintain > Apply updates to apply an update.
3. Enter a unique name for the update. You will use this name to identify the update for which you want to promote the image (both the Microsoft binary and the customer AOT package) to the production environment.
4. Select the package to apply. Use the filter at the top to find your package. The list will include application and platform binary packages and application deployable packages that have passed validation from the asset library.
5. Select Apply.

   The status in the upper-right corner of the environment details page changes from Queued to Servicing. Then, when package application is completed, the status changes to Deployed.
6. After package application is completed, the environment history is updated. To view the environment
Uninstalling a module

AOT deployable packages consist of one or many customer modules. They might be a combination of ISV modules, partner modules, or a customer’s own customization modules. If you want to completely uninstall an AOT deployable package, there are two options in the sandbox environments:

- Create a new AOT deployable package that no longer includes the module that you want to remove.
  - When you apply this package directly to your sandbox environment, a message in LCS will warn you that a module that is included in the current image of the environment is missing from your package.
    - You can proceed in LCS. Microsoft will create a new image that combines the Microsoft binary from the last update and the current AOT package that doesn't contain the module that you want to remove. In effect, the module will be uninstalled.
    - This option is advised only in situations where you don’t yet have a production environment, or where you must quickly test the resulting environment but don’t plan to promote this AOT package to production.
    - Promotion of the resulting image of the sandbox environment to production environments will be blocked.
- **Recommended option:** Use the ModuleToRemove.txt process that is outlined in Uninstall a package. This option does everything that the previous option does, but the resulting image can be promoted to production environments.

7. You can also download the logs from the environment history page.

8. After the validations are completed, you can sign-off on the update from the environment history page by selecting either Sign off or Sign off with issues.

Microsoft automatic updates in sandbox environments

On a regular basis, Microsoft will push automatic updates of new Microsoft binaries to your sandbox environments. This automatic update will be done only if your sandbox environment version has fallen behind and is older than the supported generally available version.

This automatic update will overwrite the Microsoft binary from your latest sandbox image. In effect, it will create a new update that includes the new Microsoft binary and your earlier customizations.

Promote an update to production environments

Packages are no longer applied directly to production environments. Historically, in Microsoft-managed environments, customers were able to apply any package that was successfully applied to a sandbox environment and marked as a Release candidate. However, this approach posed many challenges, because there are order of operation scenarios where application of package A before package B produced a healthy environment, but a different order led to regressing functionality.

To address these challenges, Microsoft has introduced the image-based, update process. As was discussed earlier in this topic, as packages are applied to sandbox environments, Microsoft creates images that are given an Update name value. This value represents the whole runtime, including Microsoft code and all custom code as a single unit. When customers want to promote a change to a production environment, they select an update from a sandbox environment’s history. The whole runtime is then moved to the production infrastructure as is and should better safeguard against regressions.

**IMPORTANT**

Package application causes system downtime. All relevant services will be stopped, and you won’t be able to use your environments while the package is being applied.
After you’ve successfully applied the update in the sandbox environment and are ready to move the update over to the production environment, follow these steps to mark an update as a release candidate.

1. Open the environment history page by selecting **History > Environment changes** on the environment details page.

2. Select the update to move over to the production environment.

3. In the details for the update, select **Mark as release candidate**.

   The **Is Release Candidate** option is set to **Yes**.

After you’ve marked an update as a release candidate, follow these steps to update your environment.

1. Open the environment details page for the production environment.

2. Select **Maintain > Update environment** to apply an update.

3. In the **Available sandboxes** list, select the source sandbox environment where the update was applied, validated, and marked as a release candidate.

4. In the grid, select the update to apply to the production environment. This grid shows only updates that have been marked as release candidates.

5. In the **Downtime start** field, select a date and time. The environment will be taken down for servicing at the specified time on the specified date. The **Downtime end** is calculated automatically based on the expected duration.

   No lead time is required for this update.

6. Select **Schedule**. LCS runs validations to make sure that the selected update is applicable to the environment. To prevent downgrade of the environment, the update isn’t allowed if its application version is lower than the current environment version. Additionally, customers are asked to confirm that they want the update to proceed.

   If the update is **successfully** scheduled, an email notification is sent to all project stakeholders.

   The status in the upper-right corner of the environment details page changes from **Queued** to **Servicing**. Then, when the update is completed, the status changes to **Deployed**.

   All stakeholders are notified of the progress of the operation.

7. After the update is completed, the environment history is updated. To view the environment history, select **History > Environment changes** on the environment details page.

8. You can also download the logs from the environment history page.

9. After the validations are completed, you can sign-off on the update from the environment history page by selecting either **Sign off** or **Sign off with issues**.

**NOTE**

If there is an on-going operation in the environment, or if the environment is already running on the same version or a later version, the scheduled update is canceled. When a scheduled update is canceled, an email notification is sent to all project stakeholders. Customers can also cancel an update by selecting **Cancel** on the environment details page. If customers want to reschedule or change an update, they can cancel the current operation and schedule a new one.

---

**Things to consider about production updates**

When you're promoting an update from sandbox to production, it includes **both the Microsoft binary and the customer AOT package**. Customers can't promote the Microsoft binary and the customer AOT package.
Microsoft automatic updates in production environments

If you want to promote a Microsoft binary update from your sandbox environment, be aware that it will include the latest customer packages that have been applied as of that same point in time from the sandbox.

If you want to promote a customer AOT package from your sandbox environment, be aware that it will also include the most recently applied Microsoft binary update.

In this automatic update, Microsoft won't promote any customer AOT packages. Instead, we will take the newer binary version and combine it with the latest customer AOT package in the target production environment to create a new image for the production runtime.

Rollback

For environments that are deployed in the modern infrastructure stack, if servicing is unsuccessful, the environment is automatically rolled back in most cases. To learn why the operation was unsuccessful, you can download the logs from the environment history page.

NOTE

For a small subset of environments where rollback may result in extended downtime, such as when the database size is large, the environment is left in a failed state to determine if actions can be taken to avoid performing the rollback. If the failed operation cannot move forward, then the normal rollback process is initiated.
This topic explains how you can troubleshoot issues on an environment that was deployed using the self-service deployment experience. When a user reports an issue, you can use various tools in Lifecycle Services (LCS) for troubleshooting. The rich set of telemetry data helps you build a storyboard view that shows what that user and other users were doing when the issue was reported.

To open the Environment Monitoring dashboard, follow the steps listed below:

1. Open LCS and navigate to the appropriate project.
2. In the Environments section, select the environment that you want to view, and then click Full details.
3. On the Environment details page, click Environment monitoring to open the Monitoring and diagnostics portal.

On the Environment Monitoring dashboard, you will see two tabs: Overview and Activity.

Overview tab

The Overview tab provides a storyboard view that shows how the environment was being used during a specific period. You can use the filters on this page to narrow the information logs. Here are some of the filters that are available:

- **Time duration**: Go back 60 minutes from the selected date and time.
- **User**: View a specific user’s activities.
- **Search terms**: Create a search that is based on the issue that is being investigated.
In addition, you will also see two sections:

- The **User interaction** chart shows a user’s activities on various machines in the environment and the SQL utilization trend.
- The **User activity** grid shows the various activities that users performed, based on their session timestamp. The active sessions display on the left side of the grid. For each session, the Form:Control:Command and the corresponding timestamp show when the action was taken. You can trace the exact steps that the user was taking in the information presented in this grid.

### IMPORTANT

The Overview tab will also include the CPU and memory health metrics to help with diagnostics. This feature is currently not available but will be added soon.

### Activity tab

The **Activity** tab shows a predefined set of queries for advanced troubleshooting. This gives you access to the raw information logs. You can then export the logs to do more advanced analysis. The following types of queries are available:

- User-related errors
- Slow queries
- Deadlocks
- Crashes

### NOTE

The data shown on the **Overview** and **Activity** tabs is only retained for 30 days.

### IMPORTANT

The Environment monitoring will also include advanced SQL troubleshooting tools to diagnose and mitigate performance issues. This feature is currently not available but will be added soon.
Planned maintenance is any maintenance activity that Microsoft must perform on your environments according to a published schedule. This topic provides answers to frequently asked questions about the Microsoft planned maintenance in self-service environments.

What are the types of planned maintenance activities that are performed on an environment?

Some of the common planned maintenance activities performed by Microsoft are:

- Operating system (OS) security updates
- Security hotfixes
- Microsoft package updates

What are the planned maintenance windows?

A planned maintenance window is typically during the dark hours of the geographic region that your environment is deployed in. The following list shows the maintenance windows for each geography in Coordinated Universal Time (UTC).

- Americas: 5:00 AM to 8:00 AM UTC
- EMEA: 2:00 AM to 5:00 AM UTC
- APAC: 6:00 PM to 9:00 PM UTC

What is the schedule for operating system maintenance?

<table>
<thead>
<tr>
<th>MONTH</th>
<th>AMERICAS (5:00 AM–8:00 AM UTC)</th>
<th>EMEA (2:00 AM–5:00 AM UTC)</th>
<th>APAC (6:00 PM–9:00 PM UTC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>September 2021</td>
<td>September 26, 2021</td>
<td>September 25, 2021</td>
<td>September 25, 2021</td>
</tr>
<tr>
<td>October 2021</td>
<td>October 24, 2021</td>
<td>October 23, 2021</td>
<td>October 23, 2021</td>
</tr>
<tr>
<td>November 2021</td>
<td>December 5, 2021</td>
<td>December 4, 2021</td>
<td>December 4, 2021</td>
</tr>
<tr>
<td>December 2021</td>
<td>January 9, 2022</td>
<td>January 8, 2022</td>
<td>January 8, 2022</td>
</tr>
<tr>
<td>January 2022</td>
<td>January 23, 2022</td>
<td>January 22, 2022</td>
<td>January 22, 2022</td>
</tr>
<tr>
<td>February 2022</td>
<td>February 20, 2022</td>
<td>February 19, 2022</td>
<td>February 19, 2022</td>
</tr>
</tbody>
</table>

How are operating system maintenance updates applied?

This service maintenance is planned outside normal business hours to help minimize any potential impact on your environment. For environments that have users in other parts of the world, we recognize that "outside
normal business hours” might affect you differently. We are working hard to improve Microsoft Dynamics 365 and minimize the impact of these maintenance windows in the future. Going forward, infrastructure maintenance schedules will be posted here, and you won’t receive future notifications for infrastructure maintenance.

**Can operating system updates be applied in zero downtime?**
Yes, Microsoft began to roll out near-zero-downtime infrastructure maintenance in May 2021.

**What does near-zero-downtime maintenance mean?**
Customers can continue to operate the system during the maintenance activity. They may experience brief interruptions or disconnects during this window, but will not need to take a full downtime.

**What is the experience during the near-zero-downtime maintenance window?**

**Interactive usage**
Users connected to the environment may experience a brief (up to 60 seconds) disconnect. On recovery, users may experience one of the following:

- The session recovers gracefully and the user either lands on the form they were working on or is redirected to the root/workspace/home page with the info log message “Something went wrong. But we were able to recover your session.”
- The session recovery fails and user working on a details page is redirected to the root/workspace/home page with the info log message “Something went wrong, and we were unable to recover your session. You’ve been redirected.”

For example, the user may be working on a sales order creating lines or posting. After the interruption, the user might return to the Sales workspace, but the new order and lines should still be available. We recommend that users go back to the main form and check their work.

**Batch service**
Batch service can be unavailable for up to 25 minutes. The following activities will occur:

- Any executing batch jobs will be terminated.
- Jobs that were terminated will be automatically restarted when the batch service recovers. Set the maximum number of retries to zero for any jobs that should not be restarted automatically.
  - Check printing
  - Statement posting

**NOTE**
We are working to reduce the downtime for batch service to be few minutes. This will require customers to adopt priority-based scheduling of batch jobs.

**Known issues**
The following issues are known to occur during the near-zero-downtime maintenance window:

- **Classic Material Resource Planning**: Material Resource Planning jobs are currently not recreated on restart.
  - Ensure Material Resource Planning jobs are not scheduled during the planned maintenance window. This will be fixed in a future Microsoft update.
- **Recurring integration**: Messages that were in progress at the time of restart will remain stuck in an in-
Is it possible to reschedule near-zero-downtime operating system maintenance?

No. To help with planning, the schedule for infrastructure maintenance will be published in advance every six months. Always refer to the published schedule, and be sure to plan important activities outside the operating system update window.
Microsoft provides business continuity and disaster recovery for production instances of Dynamics 365 software as a service (SaaS) applications if a Microsoft Azure region-wide outage occurs.

Customers who have purchased the appropriate licenses can deploy a production instance of a Finance and Operations app. For more information, see Cloud deployment overview.

For production environments, replicas of the different storage services (Azure SQL Database and file storage) are established in the secondary region at the time of deployment. These replicas are known as geo-secondaries.

The geo-secondaries are kept synchronized with the primary instance through continuous data replication. There is a small replication latency or lag of up to 15 minutes between the geo-secondaries and the primary instance. For more information, see Business continuity and disaster recovery (BCDR): Azure Paired Regions.

For more information about data protection in non-production environments, see Database movement operations home page.

**Planned failover**

If Microsoft determines that availability of the primary Azure region is at risk (for example, if there is an impending hurricane), we will notify customers and switch over the environment so that it operates out of the secondary region. There will be a short outage while the environment is configured for the secondary region. However, there will be no data loss, because both Azure regions are online and the replication will be caught up.

**Unplanned failover**

If an unanticipated region-wide outage occurs (for example, because of a natural disaster such as an earthquake), and Microsoft determines that the region won't become available within a reasonable amount of time, we will notify customers and switch over the environment so that it operates out of the secondary region. In this case, customers might experience data loss of up to 15 minutes, depending on the nature and timing of the outage.
Failback

When Microsoft determines that the primary region is back online and fully operational, we will notify customers and switch over the environments so that they once again operate out of the primary region. The service, including all non-production instances, is fully restored. There will be no data loss.
This topic explains how to complete the Microsoft Azure Resource Manager onboarding process for your connectors.

To deploy Azure Resource Manager topologies, you must complete the onboarding process for your connectors. To start the onboarding process, you must have the following items:

- The Azure subscription ID that you're deploying to
- Ownership of the Azure subscription, or access to the subscription owner, so that you can add contributor workflows, and the Upload Management certificate
- The tenant administrator, to work through the admin consent workflow

### Azure Resource Manager onboarding process

You can consider Azure Resource Manager onboarding a two-step procedure, where each step has its own sub-procedures. You must complete all these procedures for every subscription that you add to the Microsoft Dynamics Lifecycle Services (LCS) project.

1. Authorize the LCS deployment service to work on the Azure subscription.
   a. Authorize the workflow.
   b. Set the contributor workflow.
2. Enable the Azure subscription to deploy Azure Resource Manager resources.
   a. Enable the Azure connector, and add an LCS user.
   b. Optional: Upload the Management certificate.
   c. Configure deployment settings.

#### Authorize the LCS deployment service to work on the Azure subscription

Complete the following procedures to authorize the LCS deployment service to work on the Azure subscription.

**Authorize the workflow**

The administrator of the tenant must complete the following procedure.

1. In LCS, on the Project page, in the Environments section, click Microsoft Azure settings.
2. On the Project settings page, on the Azure connectors tab, in the Organization list group, click Authorize to start the Azure Resource Manager Contributor workflow. This workflow sets up permissions for the deployment service, so that it can deploy to your subscription on your behalf.
3. On the Grant admin consent page, click Authorize. Then sign in by using the administrator account of the Azure subscription that you must connect to, and click Accept. The authorization is now shown as completed.

**Set the contributor workflow**

Follow these steps to assign the Contributor role to the Dynamics Deployment Services [wsfed-enabled] application.

1. In the Azure portal, on the Subscription tab, select the Azure subscription, and then click the Access Control (IAM) line item.
2. Click **Add**, select **Add role assignment**. In the dialog box, set **Role** to **Contributor** and set **Assign access to** to **Azure AD user, group, or service principal**. In the **Select** field, search for and select **Dynamics Deployment Services [wsfed-enabled]**. Select **Save**.

![Add role assignment](image)

**NOTE**

Some Azure subscriptions have a **Users** section instead of an **Access control (IAM)** section. In this case, in the **Add users** dialog box, in the **Select** field, enter **Dynamics Deployment Services [wsfed-enabled]**, and then select **Select**.

3. On the **Role assignments** tab, the App is assigned as a **Contributor**.

![Add users](image)

**NOTE**

If **Dynamics Deployment Services [wsfed-enabled]** doesn't appear, the authorize process hasn't been completed, or it was completed on another Azure subscription.

**Enable the Azure subscription to deploy Azure Resource Manager resources**

Complete the following procedures to enable the Azure subscription to deploy Azure Resource Manager resources.

**Enable the Azure connector and add an LCS user**

Follow these steps to enable the Azure Connector and, as required, add an LCS user.

1. In LCS, on the **Project** page, in the **Environments** section, click **Microsoft Azure settings**.

2. On the **Project settings** page, on the **Azure connectors** tab, under **Azure connectors**, click **Add**.

![Add Azure connector](image)

**NOTE**

If you're enabling **Azure Resource Manager** for an existing connector, click **Edit**.

3. Enter the connector name, enter the Azure subscription ID to deploy to, and set the **Configure to use Azure Resource manager** option to **Yes**.

4. In the **Azure subscription AAD Tenant domain** field, enter the domain name of the Azure subscription account admin, and then click **Next**.

5. Authorize access to the subscription, either by adding the LCS user to the Azure subscription or by using
the Management certificate. **Important:** If you're adding an LCS user, continue with step 6. If you must upload a Management certificate, don't complete steps 6 through 8 of this procedure. Instead, complete the next procedure, "Upload the Management certificate."

6. In the **Azure portal**, on the **Subscription** tab, select the Azure subscription, and then click the **Access Control (IAM)** line item.

7. In the **Access Control (IAM)** dialog box, click **Add**, select **Contributor**, and then click **OK**.

8. In the **Add users** dialog box, in the **Select** field, enter the LCS user, and then press Enter.

   **NOTE**  
   You must specifically enter a user. You can't just add a group that the user is a member of. When the **Users** page opens, you can see that the user is assigned as a **Contributor**.

**Upload the Management certificate**

Complete this procedure only if you didn't complete steps 6 through 8 of the previous procedure, "Enable the Azure Connector and add an LCS user."

1. In LCS, on the **Microsoft Azure setup** page, click **Download**. Make a note of the location of the certificate file that is downloaded. You will use this information to upload the certificate to the Azure subscription.

2. In the **Azure classic portal**, in the left pane, click **Settings**.

3. Filter to the Azure subscription that is used, and then, on the **Management certificates** tab, click **Upload**.

4. Select the Management certificate that you downloaded in step 1, and then click **OK**.

**Configure deployment settings**

1. In LCS, on the **Project** page, in the **Environments** section, click **Microsoft Azure settings**.

2. On the **Microsoft Azure setup** page, select the region to deploy to, and then click **Connect**. The Azure Resource Manager onboarding flow is now completed. You should now see that the subscription has been added to the **Azure connectors** list. Additionally, a check mark should appear under **ARM Enabled**.
This topic explains how to complete the Microsoft Azure Resource Manager onboarding process for your connectors.

To deploy Azure Resource Manager topologies, you must complete the Resource Manager onboarding process for your connectors. To start the onboarding process, you must have the following items:

- The Azure subscription ID that you’re deploying to

**NOTE**
For US government Microsoft Dynamics Lifecycle Services (LCS) projects, only Azure US government–specific Azure subscriptions are supported.

- Ownership of the Azure subscription, or access to the subscription owner, so that you can add contributor workflows and upload the management certificate
- The tenant administrator, so that you can work through the admin consent workflow

**Resource Manager onboarding process**

You can consider Resource Manager onboarding a two-step procedure, where each step has its own sub-procedures. You must complete all these sub-procedures for every subscription that you add to the LCS project.

1. Authorize the LCS deployment service to work on the Azure subscription.
   
   a. Authorize the workflow.
   b. Set the contributor workflow.

2. Enable the Azure subscription to deploy Resource Manager resources.
   
   a. Enable the Azure connector, and add an LCS user.
   b. Upload the management certificate.
   c. Configure deployment settings.

**Authorize the LCS deployment service to work on the Azure subscription**

Complete the following procedures to authorize the LCS deployment service to work on the Azure subscription.

**Authorize the workflow**

Be sure that you have Azure Active Directory (Azure AD) PowerShell cmdlets installed. For more information, see [Install Azure Active Directory PowerShell for Graph](#).

The following two app IDs must be authorized on the Azure AD tenant:

- 68fdae24-7da6-4d2d-82b6-fd78a0f38eb7
- 913c6de4-2a4a-4a61-a9ce-945d2b2ce2e0

Follow these steps to authorize the app IDs on the tenant. Complete this procedure for each application.
1. Sign in via the Azure AD PowerShell cmdlet. Use the tenant administrator account to sign in.

   ```
   Connect-AzureAD
   ```

2. Make sure that the service principal isn't already installed.

   ```
   Get-AzureADServicePrincipal -Filter "AppId eq '<AppId>'"
   ```

3. Add the service principal.

   ```
   New-AzureADServicePrincipal -AppId <AppId>
   ```

4. Verify that each application ID is added successfully.

   ```
   $sp = Get-AzureADServicePrincipal -Filter "AppId eq '<AppId>'"
   ```

**Set the contributor workflow**

Follow these steps to assign the **Contributor** role to the **Dynamics Deployment Services [wsfed-enabled]** application.

1. In the **Azure portal**, on the **Subscription** tab, select the Azure subscription, and then select the **Access Control (IAM)** item on the navigation menu.

2. Select **Add**, and then select **Add role assignment**.

3. In the **Add role assignment** dialog box, set the **Role** field to **Contributor**, and set the **Assign access to** field to **Azure AD user, group, or service principal**. In the **Select** field, search for and select **Dynamics Deployment Services [wsfed-enabled]**. Then select **Save**.

**NOTE**

Some Azure subscriptions have a **Users** item instead of an **Access control (IAM)** item on the navigation menu. In this case, in the **Add users** dialog box, in the **Select** field, enter **Dynamics Deployment Services [wsfed-enabled]**. Then select **Select**.

4. On the **Role assignments** tab, the app is assigned as a contributor.
Enable the Azure subscription to deploy Resource Manager resources

Complete the following procedures to enable the Azure subscription to deploy Resource Manager resources.

Enable the Azure connector and add an LCS user

Follow these steps to enable the Azure connector and, as required, add an LCS user.

1. In LCS, on the Project page, in the Environments section, select Microsoft Azure settings.

2. On the Project settings page, on the Azure connectors tab, under Azure connectors, select Add.

   **NOTE**
   
   If you're enabling Resource Manager for an existing connector, select Edit instead of Add.

3. Enter the connector name, enter the Azure subscription ID to deploy to, and set the Configure to use Azure Resource manager option to Yes.

4. In the Azure subscription AAD Tenant domain field, enter the domain name of the Azure subscription account admin, and then select Next.

5. On the Microsoft Azure setup page, select Download. Make a note of the location of the certificate file that is downloaded. You will use this information to upload the certificate to the Azure subscription.

6. In the Azure portal, go to your subscription. In the left pane, select Management Certificates.

7. Filter to the Azure subscription that is used, and then, on the Management certificates tab, select Upload.

8. Select the Management certificate that you downloaded in step 5, and then select OK.

Configure deployment settings

1. In LCS, on the Project page, in the Environments section, select Microsoft Azure settings.

2. The Project settings page appears. In the Azure connectors area, click Add.

3. On the Microsoft Azure setup page, select the region to deploy to, and then select Connect. The Resource Manager onboarding flow is now completed. You should now see that the subscription has been added to the Azure connectors list. Additionally, a check mark should appear under ARM Enabled.
Customers can use Microsoft Azure ExpressRoute with Finance and Operations apps to connect to their on-premises infrastructure. This topic provides the information that you need to get started with ExpressRoute.

Microsoft Azure ExpressRoute lets you create dedicated, readily available, highly reliable, low latency connections between Azure datacenters and your on-premises locations. An ExpressRoute circuit is a logical connection between a customer’s on-premises network and Microsoft cloud services through a connectivity provider. ExpressRoute is configured separately from Finance and Operations apps. To get an ExpressRoute circuit for your implementation, you must contact a network service provider directly. After ExpressRoute is configured, in addition to connecting to Finance and Operations apps, customers can connect to apps such as Microsoft 365 and supported Azure services, such as connecting to virtual machines and cloud services deployed in virtual networks. To learn more about other supported services, see ExpressRoute FAQ. Before purchasing an ExpressRoute circuit, you will need to know the following information:

- The datacenter that your Finance and Operations apps are located in.
- The region where you will be connecting from.

This information is necessary to determine whether a standard or premium offering of ExpressRoute is required.

Resources for getting started

- ExpressRoute service page
- ExpressRoute technical overview
- ExpressRoute partners and peering locations
- ExpressRoute pricing
The Finance and Operations application cloud architecture contains all the elements that are common to all Microsoft cloud offerings, as described in Subscriptions, licenses, accounts, and tenants for Microsoft’s cloud offerings. Beyond this, it also includes services that automate software deployment and provisioning, operational monitoring and reporting, and seamless application lifecycle management.

The cloud architecture consists of these conceptual areas:

- **Subscription** – A subscription to Finance and Operations apps gives you an online cloud environment (or multiple environments) and experience.

- **Licenses** – Customers must purchase subscription licenses (SLs) for their organization, or for their affiliates’ employees and on-site agents, vendors, or contractors who directly or indirectly access Finance and Operations apps. These apps are licensed through Microsoft Volume Licensing and the Microsoft Cloud Solution Provider (CSP) program. For more information, download the latest Microsoft Dynamics 365 Licensing Guide from Dynamics 365 pricing.

- **Tenant** – In Microsoft Azure Active Directory (AAD), a tenant represents an organization. It’s a dedicated instance of the AAD service that an organization receives and owns when it creates a relationship with Microsoft (for example, by signing up for a Microsoft cloud service, such as Azure, Microsoft Intune, or Microsoft 365). Every AAD tenant is distinct and separate from other AAD tenants. For more information
about AAD tenants, see How to get an Azure Active Directory Tenant.

A tenant houses the company’s user information. This information includes passwords, user profile data, permissions, and related information. The tenant also contains groups, applications, and other information that pertains to an organization and its security.

The tenant is created when customers sign up for their first subscription to any Microsoft online service, such as Microsoft 365, Microsoft Dynamics 365, or Azure. Any later subscriptions to the same online services or other online services can be grouped within the same tenant.

An organization can have multiple AAD tenants. If there are multiple tenants, make sure that any subscriptions for Finance and Operations apps are associated with the correct tenant.

- **Azure Active Directory (AAD)** – AAD is the multi-tenant, cloud-based directory and identity management service from Microsoft that combines core directory services, application access management, and identity protection in a single solution. For more information, see Azure Active Directory. Finance and Operations apps use AAD as the store for identity. Access to AAD is provided as part of a subscription to Finance and Operations apps.

- **Microsoft 365 admin center** – Microsoft 365 admin center is the subscription management portal that Microsoft 365 provides for administrators. It’s used to provide management functions for users (AAD) and subscriptions. As part of these management functions, it provides information about service health. For more information, see About the Microsoft 365 admin center.

  NOTE

  You don't have to have an Microsoft 365 license to deploy Finance and Operations apps. However, you might require a license for specific Office integration scenarios. For more information, see Office integration overview.

- **Microsoft Dynamics Lifecycle Services (LCS)** – LCS is a collaboration portal that provides an environment and a set of regularly updated services that can help you manage the application lifecycle of your implementations. For more information, see Lifecycle Services resources. After you purchase and activate a subscription for a Finance and Operations app, an Implementation project workspace is provisioned in LCS when the tenant administrator signs in for the first time.

  NOTE

  An implementation project is an LCS project for the cloud service. As a Microsoft partner, you can also provision non-implementation LCS projects for your own purposes. For more information, see Lifecycle Services (LCS) for Finance and Operations apps partners.

- **Finance and Operations apps** – Finance and Operations apps are deployed through LCS. Various topologies are available: development/test/build, acceptance test, performance test, and high-availability production. For more information about the various topologies, download the latest Microsoft Dynamics 365 Licensing Guide from Dynamics 365 pricing.

- **Microsoft Azure DevOps** – Azure DevOps is used primarily for code version control, development, and to deploy a build environment. Azure DevOps is also used to track support incidents, such as work items in Azure DevOps that are submitted to Microsoft through Cloud-powered support, and to integrate the Business process modeler (BPM) library hierarchy into your Azure DevOps project as a hierarchy of work items. Azure DevOps is also used during code upgrade.

"Under the hood," Finance and Operations apps use many features of the Azure platform, such as Azure Storage, networking, monitoring, and Azure SQL Database, to name just a few. Shared services put into operation and orchestrate the application lifecycle of the environments for participants. Together, Azure functionality and LCS
offer a robust cloud service.

**NOTE**

Although many features of the Azure platform are used, you don't have to have an Azure subscription to deploy Finance and Operations apps in the Microsoft-managed cloud. You must have an Azure subscription only if you want to deploy Finance and Operations apps cloud-hosted environments in your own Azure subscription.
This topic describes how you can use Microsoft Dynamics Lifecycle Services (LCS) to automatically apply updates to cloud environments.

**IMPORTANT**

Updates are applied using deployable packages. Applying updates causes system downtime. All relevant services will be stopped, and you won't be able to use your environments while the package is being applied. You should plan accordingly.

**Supported environments**

All customer-managed and Microsoft-managed environments deployed through Lifecycle Services are supported. For more information about self-service environments, see Update an environment.

**NOTE**

If you have a build environment, you can only use LCS to apply Binary updates and Data upgrade packages. You can't use LCS to apply an Application Deployable package.

For other environments (listed below), you must use Remote Desktop Protocol (RDP) to connect to the environment and install from the command line. For information about manual package deployment, see Install deployable packages from the command line.

- Local development environments (Downloadable virtual hard disk [VHD])
- Multi-box dev/test environments in Microsoft Azure (Partner and trial projects)

**Key concepts**

Before you begin, you should understand deployable packages, runbooks, and the AXInstaller. A deployable package is a unit of deployment that can be applied in any environment. A deployable package can be a binary update to the platform or other runtime components, an updated application (AOT) package, or a new application (AOT) package. The AXInstaller creates a runbook that enables installing a package. For more details, see Packages, runbooks, and the AXUpdateInstaller in depth at the end of this topic.

**Supported package types**

- **AOT deployable package** – A deployable package that is generated from application metadata and source code. This deployable package is created in a development or build environment.
- **Application and Platform Binary update package** – A deployable package that contains dynamic-link libraries (DLLs) and other binaries and metadata that the platform and application depend on. This is a package released by Microsoft. This is available from the All binary updates tile from LCS.
- **Platform update package** – A deployable package that contains dynamic-link libraries (DLLs) and other binaries and metadata that the platform depend on. This is a package released by Microsoft. This is available from the Platform binary updates tile from LCS.
- **Commerce deployable package** – A combination of various packages that are generated after the Commerce code is combined.
• **Merged package** – A package that is created by combining one package of each type. For example, you can merge one binary update package and one AOT package, or one AOT package and one Commerce deployable package. The packages are merged in the Asset library for the project in LCS.

**NOTE**

A binary package and a Commerce deployable package can't be included in the same merged package.

For information about how to download an update from LCS and what you see in the tiles based on your environment version, see [Download updates from Lifecycle Services (LCS)](https://learn.microsoft.com/en-us/azure/lifecycle-services/downloads).

If your environment is on an application version 8.1 and later, then the **Platform Update package** does not apply to your environment. Starting with 8.1 and later releases, **Application and Platform Binary update package** is the one that applies since application and platform will be combined into a single cumulative package and will be released by Microsoft. Also note that you will no longer be applying granular X++ hotfixes and will get all application and platform updates together. This means that on the environment details page, clicking on **View detailed version information** will not have details on the granular hotfixes or KBs applied as there is no way to apply them.

**Prerequisite steps**

• **Make sure that the package that should be applied is valid.** When a package is uploaded to the Asset library, it isn’t analyzed. If you select the package, the package status appears in the right pane as **Not Validated**. A package must pass validation before it can be applied in an environment by using the following procedures. The status of the package will be updated in the Asset library to indicate whether the package is valid. We require validation to help ensure that production environments aren’t affected by packages that don’t meet the guidelines.

  There are three types of validations:
  
  - Basic package format validations
  - Platform version checks
  - Types of packages

• **Make sure that the package is applied in a sandbox environment before it’s applied in the production environment.** To help ensure that the production environment is always in a good state, we want to make sure that the package is tested in a sandbox environment before it’s applied in the production environment. Therefore, before you request that the package be applied in your production environment, make sure that it has been applied in your sandbox environment by using the automated flows.

• **If you want to apply multiple packages, create a merged package that can be applied first in a sandbox environment and then in the production environment.** Application of a single package in an average environment requires about 5 hours of downtime. To avoid additional hours of downtime when you must apply multiple packages, you can create a single combined package that contains one package of each type. If you select a binary package and an application deployable package in the Asset library, a **Merge** button becomes available on the toolbar. By clicking this button, you can merge the two packages into a single package and therefore reduce the total downtime by half.

• **Make sure that the application binary update package is applied to your dev/build environment AFTER it is applied to your sandbox and production environment** - If the application binary package is applied on your dev/build environment and this raises the platform build version to be higher than your target sandbox or production environment, you will be blocked from applying any AOT packages that are produced from this dev/build environment. To apply AOT packages produced from a dev/build environment, your dev/build instance must be equal to or lower than your target environments.
Apply a package to a non-production environment by using LCS

NOTE
For self-service type environments, see Update an environment.

Before you begin, verify that the deployable package has been uploaded to the Asset library in LCS.

1. For a binary update, upload the package directly to the Asset library. For information about how to download an update from LCS, see Download updates from Lifecycle Services (LCS). For an application (AOT) deployable package that results from an X++ hotfix, or from application customizations and extensions, create the deployable package in your development or build environment, and then upload it to the Asset library.

2. Open the Environment details view for the environment where you want to apply the update.

3. Click Maintain > Apply updates to apply an update.

4. Select the package to apply. Use the filter at the top to find your package.

5. Click Apply. Notice that the status in the upper-right corner of the Environment details view changes from Queued to In Progress, and an Environment updates section now shows the progress of the package. You can refresh the page to check the status.

6. Continue to refresh the page to see the status updates for the package application request. When the package has been applied, the environment status changes to Deployed, and the servicing status changes to Completed.

Apply a package to a production environment by using LCS

In a production environment, customers can schedule a downtime for when they want the update to be applied. For self-service type environments, see Update an environment.

IMPORTANT
An important prerequisite for applying a package to a production environment is that the package must be successfully applied to at least one sandbox environment in the same project.

1. After the update is successfully applied in a sandbox environment, go to the project's asset library. On the Asset library page, select the Software deployable package tab, select the package that you want to move to production, and click Release candidate. This indicates that this package is ready for production deployment.

2. Open the Environment details view for the production environment where you want to apply the package.

3. Select Maintain > Apply updates to apply the package.

4. Select the package to apply in your production environment, and then click Schedule to submit a request to apply it.

NOTE
The list of packages includes only the packages that have been successfully signed off in the sandbox environment, and that have been marked as release candidates.

5. Specify the date and time to schedule the package application. Click Submit, and then click OK to confirm. Note that your environments will be unavailable to perform business while the package is being
6. At the scheduled downtime, package deployment will start.

7. After the environment is serviced, you can monitor the status. The **Servicing status** field indicates the status of package application. Additionally, a progress indicator shows the number of steps that have been run, out of the total number of steps that are available.

8. After the deployment is successfully completed, the **Servicing status** field is set to **Completed**.

9. If package application isn’t successfully completed, Microsoft will investigate the issue. The **Servicing status** field will indicate that package application has failed. The environment will be rolled back to a good state.

Troubleshoot package deployment failures

If package deployment fails, see **Troubleshoot package application issues**.

Applying updates and extensions

If you are updating a Tier-2 Sandbox or Production environment on application version 8.1.2.x or newer and have initialized Cloud Scale Unit, you will also need to update Commerce channel components. For more information, see **Update Retail Cloud Scale Unit**.

If you're using components (such as Modern POS), after you've applied updates and extensions in your environment, you must also update your in-store components. For more information, see **Configure, install, and activate Modern POS (MPOS)**.

Packages, runbooks, and the AXUpdateInstaller in depth

Deployable packages, runbooks, and the AXUpdateInstaller are the tools you use to apply updates.

**Deployable package** – A deployable package is a unit of deployment that can be applied in an environment. A deployable package can be a binary update to the platform or other runtime components, an updated application (AOT) package, or a new application (AOT) package. Deployable packages downloaded from LCS or created in a development environment cannot be applied across product types. For example, a Finance deployable package cannot be applied in a Commerce app environment, and vice versa. If you have an existing customization for a Finance and Operations app that is compatible with the Commerce app, and you would like to apply it to a Commerce environment, you will need to re-package your source code in a Commerce development environment, and conversely if moving in the other direction.
Runbook – The deployment runbook is a series of steps that are generated in order to apply the deployable package to the target environment. Some steps are automated, and some steps are manual. AXUpdateInstaller lets you run these steps one at a time and in the correct order.

- Generated based on topology of deployments with multiples VMs
- Contains step by step information for applying deployable package
- Provides sequence of steps across VMs in multi-box/HA environment
- Integration for apply automation scripts at each step
  - Stop/start AOS service, batch service
  - Report deployment, DB sync, ...

AXUpdateInstaller – When you create a customization package from Microsoft Visual Studio or a Microsoft binary update, the installer executable is bundled together with the deployable package. The installer generates the runbook for the specified topology. The installer can also run steps in order, according to the runbook for a specific topology.

Additional resources
Install deployable packages from the command line
If you are updating a Tier-2 sandbox or production environment on application version 8.1.2 or newer and have initialized Commerce Scale Unit (CSU), you will also need to update channel components. This topic shows how to apply updates and extensions to CSU.

Updates to CSU are cumulative. This means that any update that you apply will include all previously released changes. Applying a Dynamics 365 Commerce deployable package for extensions is also a cumulative process and will replace the previously deployed version of the extension.

Prerequisites

Before you proceed, you must first apply updates and extensions (if applicable) to the environment. For more information, see Apply updates to cloud environments.

To update CSU, complete the following steps for each:

1. On the Environment details page, go to Environment features > Retail and Commerce.
2. On the Commerce deployment setup page, select Update.
3. In the selection panel, select the version to update to.
4. You can choose to update to the newest service update to access the latest features, or you can update to the latest quality update to apply quality improvements for the currently deployed service update. For more information, see Download updates from Lifecycle Services (LCS).
5. You can choose to apply an extension at the same time.

To apply an extension to a CSU, complete the following steps:

1. On the Commerce deployment setup page, select Apply Extension.
2. In the selection panel, select the extension to apply.

NOTE

You must first upload the Commerce deployable package to the project asset library in Microsoft Dynamics Lifecycle Services (LCS) before you can select to deploy it on the Commerce deployment setup page in LCS.

Both Apply updates and Apply extension operations will involve a period of downtime that may last up to 1 hour, or in some cases up to 2 hours or more. For example, when updating non-US locations of CSU, large data volumes, or complex schema updates. For a realistic estimate of the downtime duration, note the downtime duration in Sandbox UAT for an equivalent update and dataset that you plan to use in your production environment. During this time, the following will occur:

- Cloud-hosted Commerce channels will not function (unless POS offline capability is enabled).
- POS devices that have the offline capability feature enabled will have reduced functionality.
- Any e-commerce clients that are dependent on CCSU will be disrupted.
- Channels hosted on CSUs will remain unaffected.
- Head office functionality will remain unaffected.
View history

To view the history of recent operations on a Scale Unit, select History on the Action tab to open the Scale Unit History page. On this page, you can view recent operations such as initialize, service update, quality update, version, extension details, and other relevant information.

Restart Commerce Scale Unit

For troubleshooting, Commerce Scale Unit allows self-service restart of the service. Commerce Scale Unit restart can be useful for mitigation of service reliability or performance issues. To restart Commerce Scale Unit, select a Scale Unit and then select Restart. A dialog box will display. Select Restart to restart the service. Restarting Commerce Scale Unit will close all active connections before restarting. You can also choose an immediate restart by selecting the Force restart option. This will immediately close all active connections and initiate a restart.

CSU auto-update sequence

Auto-update for CSU is being gradually rolled out to all Commerce customers. If you are a LCS project owner or environment administrator, you'll receive an email notification when CSU auto-update is rolled out to your LCS project.

When CSU is auto-updated by Microsoft, it takes place in the following sequence.

CSU Auto-update sequence
This feature currently applies to cloud-hosted Commerce Scale Unit. Self-hosted Commerce Scale Unit is not included and must be self-updated.

Auto-update for Commerce Scale Unit enables One Version auto-update. All existing One Version processes, policies, and schedules apply to Auto-update for Commerce Scale Unit.

NOTE
Auto-update for Commerce Scale Unit is being incrementally rolled-out to customers and is expected to be available to all customers by end of 2021.

Limitations
The following limitations currently exist and are planned to be resolved in upcoming updates:

- In-app notifications are not available.
- If you have multiple Commerce Scale Units in a sandbox UAT environment, Commerce Scale Unit will only be auto-updated based on the first Commerce Scale Unit in that environment (alphabetically). The remaining Commerce Scale Units in each sandbox UAT environment will need to be self-updated.
- Auto-update for Commerce Scale Unit is not currently available for First Release customers, and is not currently applicable for Preview Early Access Program (PEAP).

Downtime duration and impact
Updates to Commerce HQ and Commerce Scale Unit (cloud) are applied sequentially. Downtime duration is typically one hour, but varies by data volume and region. To estimate downtime duration in your production environment, you can either self-update a Commerce Scale Unit in your sandbox UAT environment, or review the total update duration for both Commerce HQ and Commerce Scale Unit in Lifecycle Services (LCS).

NOTE
Downtime duration will vary for each update and data volume. To estimate a realistic downtime duration, ensure that your sandbox UAT environment has the same data as your production environment. To do this, follow the steps in Refresh database. You also need to apply the same update in the sandbox UAT environment that you plan to estimate the downtime duration for.

For retailers with a business need for redundancy, Modern POS offline capability allows core POS operations to be available for use while disconnected from the internet or while the cloud environment is being updated. Stores operating with Commerce Scale Unit will also continue to operate with support for core POS operations during cloud maintenance windows. For more information, see Online and offline point of sale (POS) operations.

Version support and backward compatibility
All in-store components must be running released software that is less than 12 months old in order to maintain support. Customers are responsible for updating self-hosted components (such as components
Updating in-store components

You can choose to update self-hosted components manually at each store or use mass update tools such as Microsoft System Center Configuration Manager or Microsoft Intune.

With auto-update for Commerce Scale Unit, updates to in-store components can still be rolled-out in phases. The following controls are available to achieve this:

- **Screen layout designer** – Most visual elements in POS are configured and centrally managed by an administrative user in the customer organization. This means that new POS operations will not automatically be displayed on POS unless explicitly configured for inclusion in corresponding screen layouts. Screen layouts are configured using Screen layout designer and can be specific to a store or POS device. For more information, see [POS user interface visual configurations](#).

- **Functionality profiles, POS permissions, Commerce parameters** – Many elements of POS functionality are typically configurable by the user. This can be configured through functionality profiles, POS permissions, Commerce parameters, or other controls which allow for device, register, store, or user-level functionality control in applicable scenarios.

- **Modern Point of Sale and Commerce Scale Unit** – Because Modern Point of Sale and Commerce Scale Unit are self-hosted by the retailer, topologies which include either of these components enable roll out of updates at a separate (and slower) cadence, and in a more granular fashion than with cloud-only topologies.

- **Feature flags** – Feature flags, configurable in Commerce HQ, provide additional controls to selectively turn features on/off.
If you’re using a Tier-2 sandbox or production environment that has application version 8.1.2.x or later, you must initialize Commerce Scale Unit (cloud) before you can use retail channel functionality either for point of sale (POS) operations or for e-commerce operations that use Retail Server in the cloud. Initialization will deploy a Commerce Scale Unit (cloud).

This topic describes the steps for initializing Commerce Scale Unit (cloud).

**IMPORTANT**

For existing customers using retail channel functionality in the cloud, to ensure continued and uninterrupted support for your business, we require that you update your retail channels to use Commerce Scale Unit. New environments deployed without Commerce Scale Unit will no longer receive quality and service updates for cloud-hosted retail channel components. There is no action required for customers who exclusively use Commerce Scale Unit (self-hosted). Contact your Microsoft FastTrack solution architect if you require an extension.

**Prerequisites**

1. Deploy a Tier-2 sandbox or production environment that has version 8.1.2.x or later.
2. You can self-deploy up to 2 Commerce Scale Units per environment. If you require more than 2 Commerce Scale Units per environment, in Microsoft Dynamics Lifecycle Services (LCS), create a support request, and enter *Request for additional Commerce Scale Unit* and indicate the environment ID, number of Commerce Scale Units, and desired datacenter regions. The request will be completed within five business days. If you do not require more than 2 Commerce Scale Units per environment, you do not need to create a support request.
3. You must have Project Owner permissions in Lifecycle Services before you can initialize Commerce Scale Unit.
4. Ensure that Retail license configuration keys are enabled in your environment. For more information, see [License codes and configuration keys report](#). You must have the following keys turned on to use Commerce Scale Unit.
   - RetailBasic
   - RetailECommerce - If you plan to use E-Commerce for Dynamics 365 Commerce.
   - RetailGiftCard - If you plan to use gift cards.
   - RetailInvent - If you plan to use inventory.
   - RetailModernPos - If you plan to use point of sale (POS).
   - RetailReplenishment - If you plan to use replenishments.
   - RetailScheduler
   - RetailStores - If you plan to use POS.

**Region availability**

Commerce Scale Unit is available for deployment in the following regions.
<table>
<thead>
<tr>
<th>GLOBAL LOCATION</th>
<th>REGION</th>
<th>AVAILABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMERICAS</td>
<td>East US</td>
<td>Generally available</td>
</tr>
<tr>
<td>AMERICAS</td>
<td>East US 2</td>
<td>Generally available</td>
</tr>
<tr>
<td>AMERICAS</td>
<td>North Central US</td>
<td>Generally available</td>
</tr>
<tr>
<td>AMERICAS</td>
<td>South Central US</td>
<td>Generally available</td>
</tr>
<tr>
<td>AMERICAS</td>
<td>Central US</td>
<td>Generally available</td>
</tr>
<tr>
<td>AMERICAS</td>
<td>West US</td>
<td>Generally available</td>
</tr>
<tr>
<td>AMERICAS</td>
<td>West US 2</td>
<td>Generally available</td>
</tr>
<tr>
<td>AMERICAS</td>
<td>Canada Central</td>
<td>Limited capacity</td>
</tr>
<tr>
<td>AMERICAS</td>
<td>Canada East</td>
<td>Limited capacity</td>
</tr>
<tr>
<td>AMERICAS</td>
<td>West Central US</td>
<td>Limited capacity</td>
</tr>
<tr>
<td>APAC</td>
<td>Australia East</td>
<td>Generally available</td>
</tr>
<tr>
<td>APAC</td>
<td>Southeast Asia</td>
<td>Generally available</td>
</tr>
<tr>
<td>APAC</td>
<td>Japan East</td>
<td>Generally available</td>
</tr>
<tr>
<td>APAC</td>
<td>Japan West</td>
<td>Generally available</td>
</tr>
<tr>
<td>APAC</td>
<td>Australia Southeast</td>
<td>Limited capacity</td>
</tr>
<tr>
<td>APAC</td>
<td>East Asia</td>
<td>Limited capacity</td>
</tr>
<tr>
<td>APAC</td>
<td>India South</td>
<td>Limited capacity</td>
</tr>
<tr>
<td>APAC</td>
<td>India Central</td>
<td>Limited capacity</td>
</tr>
<tr>
<td>EMEA</td>
<td>West Europe</td>
<td>Generally available</td>
</tr>
<tr>
<td>EMEA</td>
<td>North Europe</td>
<td>Generally available</td>
</tr>
<tr>
<td>EMEA</td>
<td>UK South</td>
<td>Limited capacity</td>
</tr>
<tr>
<td>EMEA</td>
<td>UK West</td>
<td>Limited capacity</td>
</tr>
</tbody>
</table>

Deployment capacity in Limited capacity regions are extremely constrained. Requests for deployment are evaluated on a case-by-case basis. If you have a compelling business need for deployment in Limited capacity regions, you can file a support request to be added to the waitlist.
Initialize Commerce Scale Unit as part of a new environment deployment

Please make sure the headquarters is available. This is required to register the scale unit with the headquarters during the initialization process. It is not recommended to initialize a scale unit when the headquarters is under servicing, as it may become unavailable during its servicing process.

1. Make sure the headquarters environment is available and not in Maintenance mode.
2. In LCS, on the environment details page, select Environment features > Commerce.
3. On the Commerce setup deployment page, select Initialize.
4. Select the version of the Commerce Scale Unit to initialize.
5. Select a region to initialize Commerce Scale Unit in.

Configure retail channels to use Commerce Scale Unit

1. After Commerce Scale Unit has been deployed, in the head office client go to Retail and commerce > Retail Headquarters > Retail Scheduler setup > Channel database to ensure that your retail channels are configured to use the database for this Commerce Scale Unit.
2. Go to each retail channel and select the Channel Profile for the corresponding Commerce Scale Unit.

Database refresh and Commerce Scale Units

Before you begin, make sure you are familiar with Steps to complete after a database refresh for environments that use Commerce functionality.

The scale unit channel database records (in the Channel Database form) cannot be moved across environments as part of database refresh. This is because the records represent environment specific configuration.

After database refresh, you can regenerate the scale unit’s channel database record by issuing a re-deployment of your scale unit in LCS. Any deployment or servicing operation in the scale unit will attempt to register the scale unit with the headquarters, if the registration is detected as missing.

You can issue a re-deployment of the scale unit, without changing any components, by selecting to deploy the same version your scale unit is at already. This can be done in LCS by the following steps:

1. In LCS, on the environment details page, select Environment features > Retail.
2. On the setup deployment page, select the scale unit you would like to redeploy.
3. On the scale unit’s operation menu, select **Update**.
4. On the slider, on the drop-down for **Select version**, pick the option **Specify a version**.
5. On the text box under **Specify a version**, type in the version shown for your scale unit, shown besides the **Current version** label.
6. Click on **Update** button.

You do not need to select **Update extensions**, even if you have applied extensions previously, since the last extension package applied to the scale unit is automatically picked when updating a scale unit.

If you have multiple scale units, you need to perform the operation above for each scale unit. You may perform these operations in parallel, if desired.

**Deploy additional Commerce Scale Units (optional)**

After you have initialized the first Commerce Scale Unit (CSU), if you require additional cloud scale units, enter a support request. In the support request, state the number of RCSUs needed, environment name, and desired regions.

For each additional RCSU that you deploy, it is also recommended that you create a separate channel database group for each RCSU. To do this, follow these steps:

1. In Commerce head office, go to **Retail and commerce > Retail Headquarters > Retail Scheduler setup > Channel database group**.
2. Create a new channel database group.
3. Go to the **Retail and commerce > Retail Headquarters > Retail Scheduler setup > Channel database** form and select the channel database that corresponds to the newly created RCSU.
4. Select **Edit** and select the new channel database group.
5. Select **Save**.
6. Select **Run Full data sync** for the selected channel database.

**Additional considerations if you initialize cloud-hosted Commerce channel components in an existing environment**

If you’re already using cloud-hosted Commerce channel components in an environment, initialization of Commerce Scale Unit will help reduce the downtime when those components are updated. Additional planning is required before initialization of Commerce Scale Unit.

When you initialize your first Commerce Scale Unit in an environment that uses cloud-hosted Commerce channel components, the initialization process will migrate your channels associated to the cloud-hosted channel components to the first scale unit. Channels associated with Store Scale units are unaffected.

The migration process is transparent to the channels. After the scale unit initialization starts, the following operations are automatically performed:

1. A new Commerce Scale Unit will be created and associated with the environment.
2. The Commerce Scale Unit will be registered as an available Channel Database in the headquarters.
3. All channels mapped to the **Default** channel database in the headquarters will be updated to map to the new Commerce Scale Unit.
4. A Commerce Data Exchange (CDX) full data sync will be performed to bring the channel data to the new scale unit.

**Planning and testing for Commerce Scale Unit initialization** As a general rule, when initializing Commerce Scale Unit, you must plan for a five-hour downtime window for store operations as well as any e-
commerce channel operations that use Retail Server or Cloud Point of Sale.

1. Perform a database refresh from your production environment to a sandbox UAT environment.
2. Initialize Commerce Scale Unit in the sandbox UAT environment.
3. Note the initialization time to complete for Commerce Scale Unit. This will be comparable to the time this operation takes in your production environment, during which store operations and e-commerce operations will be unavailable.

You must perform the following additional steps before initializing Commerce Scale Unit.

- **Close all POS shifts** - After migration, POS users will be unable to close any shifts that were active during the migration process. 
- **Validate that all P-jobs have been successfully completed** - It is recommended that P-jobs to synchronize pending transactions have completed before CSU is initialized.
- **Sign out of all POS device** - POS operations are not supported during migration.
- **Recall and void all suspended transactions at POS** - Suspended transactions are not preserved as part of the initialization.

As part of Commerce Scale Unit initialization, prior suspended transactions will be lost and cannot be recalled.

Here is what occurs during the initialization period:

- Cloud-hosted Commerce channels won't work, unless you turn on POS offline capability.
- POS devices with offline capability turned on will have reduced functionality.
- Any e-Commerce clients that depend on Retail Server will be disrupted.
- Channels that are hosted on Commerce Scale Units (self-hosted) won't be affected.
- Head office functionality is not affected.

Here is what occurs after initialization is completed:

- The device activation state of all activated POS devices is preserved, which means that the devices won't have to be reactivated.
- Stand-alone hardware station instances will continue to work.
- POS channel–side reports will be reset and won't show data from before the initialization.
- Show journal operation will also be reset and won't show data from before the initialization.
This topic explains how to migrate Microsoft Dynamics 365 Commerce store channels from the Commerce Scale Unit (CSU) that they are currently working with to a different CSU. You might want to migrate channels to a different CSU for better load isolation and resource governance between channels, to reduce latency to your stores, or to manage different update/extension deployment schedules for staged roll-out and pilots. Migration to a different CSU involves downtime for the channels.

This topic describes best practices that will help you minimize business disruption and downtime while you migrate channels. It applies to the migration of channels between cloud-hosted CSUs, between self-hosted CSUs, from cloud-hosted CSUs to self-hosted CSUs, and from self-hosted CSUs to cloud-hosted CSUs.

**NOTE**

If you migrate channels between CSUs, temporary sales data that was used for journal records and point of sale (POS) reports before the migration will no longer be available at the POS after migration. After the migration is completed, journals and channel reports will be started afresh by using new data.

In the following procedures, the terms *origin* and *destination* are used to distinguish the CSUs and corresponding channel databases that are involved in the migration.

### Planning for downtime

When you follow the procedures that are described in this topic, all long-running system processes that are involved are run before the actual migration, while the stores are still operational. These processes including synchronization of master data for products, prices, and customers. Then, during the migration, the critical period when you must take planned downtime in your environment involves data synchronization of a very small payload of channel configuration data to the new CSU. In most cases, this synchronization can be completed in under 10 minutes. However, from an operational perspective, you must plan for a longer downtime window to ensure that all prerequisite steps have enough time to be completed. These steps include closing all shifts, syncing transactions to Commerce headquarters, and posting statements. The amount of time that is required will vary by organization.

### Prerequisites

First, complete all the following procedures in a sandbox user acceptance testing (UAT) environment. Then repeat them in your production environment. In this way, you can measure and estimate the downtime that you should expect in the production environment.

### Migration

**Configure data synchronization**

The following steps can be completed while the stores are still operational. They synchronize master data to the destination CSU.

1. In Commerce headquarters, on the Channel database page, select the record for the channel database for the destination CSU.
2. Add the desired channels to the destination channel database.
3. Select **Full data sync**, and specify that job **9999 (All jobs)** should be used.

```markdown
NOTE
If your destination CSU is self-hosted, consider creating separate channel database groups to reduce the volume of unnecessary master data synchronization.
```

**Prepare for migration**

This procedure and the next procedure must be completed during the planned downtime for your channels.

1. On all POS devices, make sure that all shifts are closed.
2. Sign out of all POS devices.
3. Confirm that all POS offline transactions are synced to Commerce headquarters. Run P-jobs for all existing channel databases that will be migrated. If you start the migration before P-jobs are completed, and if you use Cloud POS, you might lose transactions because of duplicate transaction numbers.
4. Confirm that all statements are posted.

**Migrate channels to a new CSU**

1. On the **Store details** page, set the **Live channel database** field to the destination CSU database.
2. Set the **Channel profile** field to the channel profile that is associated with the destination CSU.
3. On the **Distribution schedule** page, run job **1070 (Channel configuration job)** and job **1110 (Global configuration job)**. Select **Run now** for each job. (For asynchronous processing, select **Create batch job** instead of **Run now**.) After the jobs are completed, your channels have been migrated to the new CSU.
4. If you're using Cloud POS, you must use the URL of Cloud POS for the new CSU and reactivate the POS device. If you reactivate the POS device before all P-jobs on the origin channel database are completed, you might lose transactions because of duplicate transaction numbers.

   If you're using Modern POS, close each POS device, reopen it, and sign in.

**Post-migration**

You can continue to use the origin CSU to serve other channels.

```markdown
NOTE
Do not delete the origin CSU. Doing so may make the store unoperable.
```

After you've completed all the procedures in a sandbox UAT environment, repeat them in your production environment.
Choose a region

When you initialize a Commerce Scale Unit (cloud), you need to select a data center location to host. To minimize network latency and improve performance, you should choose a datacenter location that is in proximity to the channels that you plan to serve using the RCSU. To reference approximate locations of each datacenter, see Azure regions. You can also reference a web-based utility, such as Azure speed reference, and measure latency to Azure datacenters from each store location. This can help you to make the right choice of datacenter when you initialize the RCSU.

Data between regions

If you initialize RCSU in a data center that is different than where your head office is located, the data will travel between these data centers with periodic synchronization. The system is pre-configured to transfer specific types of data. You can modify this configuration to synchronize different data.

To view the data synchronization configuration, go to Retail and Commerce > Headquarters setup > Commerce scheduler > Scheduler jobs to view the data synchronization jobs and sub-jobs. To view the fields being synchronized, click through a sub-job.

Synchronize specific segments of records

You can configure Commerce Data Exchange (CDX) so that only specific segments of records are synchronized to specific RCSUs.

Prerequisites

Before you configure the record segments that will be synchronized, you need to configure Channel database groups. These database groups must be configured for each CSU where you want to synchronize segmented data. All CSUs in the same Channel database group receive the data that is needed to serve all the channels in that CSU. You will need to create a separate database group for each CSU channel database where you plan to synchronize segmented data. To do this, perform the following steps:

1. Go to Retail and Commerce > Headquarters setup > Commerce scheduler > Channel database group.
2. Create a new database group for each CSU where you want to synchronize segmented data.
3. Go to Retail and Commerce > Retail and Commerce IT > Distribution schedule.
4. Add the newly created Channel database group to each scheduler job.
5. Go to Retail and Commerce > Headquarters setup > Commerce scheduler > Channel database.
6. Select the Channel database for the corresponding Commerce Scale Unit (cloud).
7. Select Edit and choose the channel database group that you created in step 2.
8. Select Save.
9. Select Full data sync for all jobs for the selected channel database.

Available controls for synchronizing segments of data to different RCSUs

Channel configuration data will only be synchronized for channels that are served by specific RCSUs. For example, only the channel configuration data for channels that are configured to be served by the West Europe channel will be synchronized to that RCSU.
Using assortments, product and pricing related data can be segmented by specific channels. For example, if your stores in North America only sell women's shoes and your stores in West Europe only sell sporting goods, you can use Assortments to segment this data. When data is synchronized to RCSU, women's shoes data will only be synchronized to the North America RCSU and sporting goods data will only be synchronized to the West Europe RCSU.

Customer and employee records are configured using the Global Address Book, which can be configured for specific channels. For example, you can configure separate address books for customers and employees for channels served by the West Europe RCSU and those for North America RCSU.
You can deploy Dynamics 365 Finance + Operations (on-premises). When you choose an on-premises deployment type, the system requirements, hardware sizing, and functionality differ from a cloud deployment. This topic provides links to content that contains information specific to on-premises deployments.

Get started

- On-premises deployment overview
- Plan and prepare for on-premises deployments
- System requirements for on-premises deployments
- Microsoft Dynamics 365 Finance + Operations (on-premises) supported software
- Hardware sizing requirements for on-premises environments
- Buy Finance + Operations (on-premises)
- Comparison of cloud and on-premises features

Onboard

- Set up on-premises projects in Lifecycle Services (LCS)
- Set up and deploy on-premises environments (Platform update 41 and later)
- Set up and deploy on-premises environments (Platform updates 12 through 40)
- Install network printer devices in on-premises environments
- Configure SQL Server Reporting Services for on-premises deployments
- Develop and deploy custom models to on-premises environments

Work in your on-premises deployment

- Configure high availability for SQL Server Reporting Services (SSRS) nodes
- Configure document management
- Import Electronic reporting (ER) configurations
- Document generation, publishing, and printing in on-premises deployments
- Configure proxies for on-premises environments
- Set up technical support for Finance and Operations apps
- Client internet connectivity
- Apply updates to on-premises deployments
- Redeploy on-premises environments
- Reuse the same AD FS instance for multiple environments

Commerce

- Commerce capabilities that are available in on-premises deployments
- Installation steps for Retail channel components in an on-premises environment
- Configure, install, and activate Modern POS (MPOS)
- Configure and install Commerce Scale Unit
Upgrade

- In-place upgrade process for on-premises environments

Other resources

- Troubleshoot on-premises deployments
- Scripts for resolving issues in on-premises environments
- Certificate rotation
- On-premises diagnostics
- Removed or deprecated features for Finance and Operations
- Software lifecycle policy and on-premises releases
Microsoft Dynamics 365 Finance + Operations (on-premises) supports running business processes in customer data centers. With this deployment option, application servers and the Microsoft SQL Server database will run in the customer’s data center. Customers and partners will utilize Microsoft Dynamics Lifecycle Services (LCS) to manage their on-premises deployments. LCS is an application management portal that provides tools and services for managing the application lifecycle of your implementations in the cloud and on-premises. LCS features, such as business process modeling, software deployment and patching, and monitoring and diagnostics, are used to help support on-premises deployments.

IMPORTANT
Dynamics 365 Finance + Operations (on-premises) is not supported on any public cloud infrastructure, including Microsoft Azure Cloud services. However, it is supported to run on Microsoft Azure Stack Hub services.

Architecture

The on-premises deployment option uses cloud components running on-premises using Microsoft Azure Server Service Fabric standalone clusters. Service Fabric is the next-generation Microsoft middleware platform for building and managing enterprise-class high-scale applications. Service Fabric standalone clusters can be deployed on any computer that is running Windows Server.

On-premises deployment defines two types of Service Fabric standalone clusters: clusters for production environments and clusters for sandbox environments. The following roles or node types are deployed into both types of clusters:

- Application Object Servers (AOS) – Provides the ability to run the application functionality in client, batch, and import/export scenarios.
- Management Reporter (MR) – Provides financial reporting functionality.
- SQL Server Reporting Services (SSRS) – Provides document reporting functionality.
- Environment Orchestrator – Enables on-premises environment management from LCS.

Figure 1 shows a logical diagram of the node types deployed in a Service Fabric standalone cluster.

Application lifecycle management for on-premises deployments is orchestrated through LCS. Customers can use the proven tools and methodologies in LCS to help manage their on-premises deployments (Figure 2).
Data storage

The on-premises deployment option stores core customer data on-premises. Core customer data is a subset of the customer data definition provided in the Microsoft Trust Center. Table 1 outlines the categories of customer data that are stored in Microsoft Azure data centers located in the United States by services such as LCS, Azure Active Directory, and Microsoft Office signup portal. All other customer data, referred to as core customer data, is stored on-premises.

Table 1: Customer data stored in Microsoft Azure data centers located in the United States by services supporting on-premises environments. These services enable initial onboarding, initiation, and tracking of support incidents, and service updates and upgrades.

<table>
<thead>
<tr>
<th>SUPPORTING SERVICES</th>
<th>CUSTOMER DATA DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Dynamics Lifecycle Services</td>
<td>Project content and files are stored in a project. This includes application configuration data, code, metadata, and data assets that comprise the application and business process models. Also included is anonymized user activity logs and information that is collected during the onboarding process.</td>
</tr>
<tr>
<td>Microsoft Office signup portal</td>
<td>Customer information that is collected during the onboarding process.</td>
</tr>
<tr>
<td>Microsoft Azure Active Directory</td>
<td>Authentication for LCS and Azure DevOps.</td>
</tr>
</tbody>
</table>

Additional services or components can be configured to extend an on-premises deployment as needed; however, configuration choices may cause core customer data to be transferred outside of the customer’s data center. For example, configuring data management features that are used to integrate external services with an on-premises deployment may result in the transfer of core customer data outside the on-premises deployment.
Dynamics 365 Finance + Operations (on-premises) supports running business processes in customer data centers. With this deployment option, application servers and the Microsoft SQL Server database will run in the customer’s data center.

This topic will help you plan and prepare for your on-premises deployment.

**IMPORTANT**

Dynamics 365 Finance + Operations (on-premises) is not supported on any public cloud infrastructure, including Microsoft Azure Cloud services. However, it is supported to run on Microsoft Azure Stack Hub services.

Differences between cloud deployments and on-premises deployments

The features in cloud deployments and on-premises deployments differ. These differences will affect your planning. The differences are described in the following topics:

- Deployment options
- Comparison of cloud and on-premises features
- Removed or deprecated features for Finance and Operations

How LCS is used with on-premises deployments

Microsoft Dynamics Lifecycle Services (LCS) is an application management portal that provides tools and services for managing the application lifecycle. Customers and partners use LCS to manage both cloud and on-premises deployments. You can use LCS for the following tasks:

- Deploy cloud and on-premises environments.
- Service your environments.
- Monitor, diagnose, and analyze the health of the environments that you manage (cloud only).
- Search for product issues and regulatory features.
- Obtain support.

For more information about LCS, see Lifecycle Services resources.

Environments

There are four types of environments that you need to plan for. This section describes the four environments and how to access and deploy them.

**Demo environment**

You can sign up for a demo environment to learn about the system. The demo environment is applicable to both cloud and on-premises deployments. For more information, see Sign up for preview subscriptions.

**Developer environment**

The development experience is the same for cloud and on-premises deployments. To access a developer environment, see Deploy and access development environments.
**Sandbox environment**

Business users and functional team members validate application functionality by using a sandbox environment. This functionality includes customizations and data that was brought forward from Microsoft Dynamics AX 2012 environments. To deploy an on-premises sandbox environment, see Set up and deploy on-premises environments home page.

At a minimum, an on-premises sandbox environment requires:

- 3 machines running Environment Orchestrator
- 2 machines running Application Object Servers (AOS)
- 1 machine running Management Reporter (MR)
- 1 machine running SQL Server Reporting Services (SSRS)
- 1 machine running Active Directory
- 1 machine running SQL Server

**Production environment**

The production environment is the live deployment that your users and customers have access to. To deploy a production environment, see Set up and deploy on-premises environments home page.

At a minimum, an on-premises production environment requires:

- 3 machines running Environment Orchestrator
- 3 machines running Application Object Servers (AOS)
- 1 machine running Management Reporter (MR)
- 1 machine running SQL Server Reporting Services (SSRS)
- 2 or more machines running SQL Server
- 2 or more machines running Active Directory

**Service Fabric**

An on-premises deployment uses Azure Service Fabric standalone clusters. Service Fabric is the next-generation Microsoft middleware platform for building and managing enterprise-class, high-scale applications. Service Fabric standalone clusters can be deployed on any computer that is running Windows Server.

An on-premises deployment has a standalone cluster for each sandbox environment and a standalone cluster for each production environment. The following roles or node types are deployed into both types of clusters:

- Application Object Servers (AOS) – Provides the ability to run the application functionality in client, batch, and import/export scenarios.
- Management Reporter (MR) – Provides financial reporting functionality.
- SQL Server Reporting Services (SSRS) – Provides document reporting functionality.
- Environment Orchestrator – Enables on-premises environment management from LCS.

The following diagram shows the node types deployed in a Service Fabric standalone cluster.
Service Fabric resources

To learn more about Service Fabric, see the following topics:

- Azure Service Fabric documentation - To learn more about Service Fabric.
- Service Fabric application upgrade - An Azure Service Fabric application is a collection of services that requires periodic upgrades.
- Plan and prepare your Service Fabric standalone cluster deployment - Additional information about Service Fabric clusters and antivirus exclusions.

System requirements

Review the system requirements in System requirements for on-premises deployments and be aware of the number of machines that are required for on-premises deployments.

Hardware sizing

Before you begin the hardware and infrastructure sizing process for an on-premises environment, familiarize yourself with the System requirements for on-premises deployments and Set up and deploy on-premises environments home page to gain a solid understanding of the underlying infrastructure. Pay close attention to the system setup best practices for optimum performance. After you have reviewed the documentation, you can start the process of estimating your transactional and concurrent user volume and sizing your environment based on the average core throughput.

Factors that affect sizing

The core factors that affect sizing are:

- Transaction characterization
- User characterization - Type and concurrency
- Data composition
- Extensions
- Reporting usage patterns
- Third-party solutions

The more detailed data that you collect, the more precisely you can estimate sizing. Hardware sizing, without supporting data, is likely to be inaccurate. The minimum data that you need to collect is the peak transaction line load per hour. The factors that affect sizing are shown in the following diagram.
From left to right, the first and most important factor needed to accurately estimate sizing is a transaction profile or a transaction characterization. It’s important to find the peak transactional volume per hour. If there are multiple peak periods, then these periods need to be accurately defined.

As you understand the load that impacts your infrastructure, you also need to understand more detail about these factors:

- **Transactions** – Transactions typically have certain peaks throughout the day or week. The peaks might depend on the transaction type. For example, time and expense entries usually show peaks once per week, while sales orders might arrive in bulk via integration or trickle in during the day.

- **Number of concurrent users** – The number of concurrent users is the second most important sizing factor. You cannot get reliable sizing estimates based only on the number of concurrent users. If concurrent users is the only data that you have available, then estimate an approximate number for transactions, and revisit this when you have more data. An accurate concurrent user definition means that:
  - Named users are not concurrent users.
  - Concurrent users are always a subset of named users.
  - Peak workload defines the maximum concurrency for sizing. For concurrent users, the user must meet all the following criteria:
    - The user is logged on.
    - There are working transactions or inquiries at the time of counting.
    - The session is not idle.

- **Data composition** – Data composition is how your system will be set up and configured. For example, this can include the number of legal entities, the number items, the number of BOM levels, and how complex the security setup will be. Each of these factors might have an impact on performance, however the impact can be offset by using smart choices when it comes to infrastructure.

- **Extensions** – Customizations can be simple or complex. The number of customizations and the nature of complexity and usage has a varied impact on the size of the infrastructure needed. For complex customizations, you should conduct performance evaluations to ensure that they are not only tested for efficiency but also help understand the infrastructure needs. This is even more critical when the extensions are not coded according to best practices for performance and scalability.

- **Reporting and analytics** – Reporting and analytics typically include running heavy queries against the database systems. Reducing the frequency of when data intensive reports run will help reduce their impact. It’s also important to understand how the design of your queries impacts their performance.
• **Third-party solutions** – These solutions, like ISVs, have the same implications and recommendations as extensions.

**Sizing your environment**

To determine your sizing requirements, you must know the peak volume of transactions that you need to process. Most auxiliary systems, like Management Reporter or SSRS, are less mission critical. As a result, this topic focuses primarily on AOS and SQL Server.

In general, the compute tiers scale out and should be set up in an N+1 fashion, meaning if you estimate three AOS, add a fourth AOS. The database tier should be set up in an Always On highly-available setup.

**SQL Server (OLTP)**

**Sizing**

• 3K to 15K transaction lines per hour per core on DB server.

• Typical AOS-to-SQL core ratio 3:1 for the primary SQL Server. Additional cores are required based on the high-availability configuration.
  o Processing database-heavy operations may regress this to 2:1.

• The following factors influence variations:
  o Parameter settings in use.
  o Levels of extensions.
  o Additional functionality usage, such as database logs and alerts. Extreme database logging will further reduce throughput per hour per core below 3K lines.
  o Complexity of data composition. For example, a simple chart of accounts versus a detailed fine-grained chart of accounts has implications on throughput.
  o Transaction characterization.
  o 2 GB to 4 GB memory for each core.
  o Auxiliary databases on DB server such as Management reporter and SSRS databases.
  o Temp DB = 15% of DB size, with as many files as physical processors.
  o SAN size and throughput based on total concurrent transaction volume/usage.

**High availability**

You should always utilize SQL Server in either a cluster or mirroring setup. The second SQL node should have the same number of cores as the primary node.

**Active Directory Federation Services (AD FS)**

For AD FS sizing, see the [AD FS Server Capacity documentation](#). A sizing spreadsheet is available for planning the number of instances in your deployment.

**AOS (Online and batch)**

**Sizing**

• Sizing by transaction volume/usage
  o 2K to 6K lines per core
  o 16 GB per instance
  o Standard box – 4 to 24 cores
  o 10 to 15 Enterprise users per core
  o 15 to 25 Activity users per core
  o 25 to 50 Team members per core

• Batch
  o 1 to 4 batch threads per core
  o Size based on batch window characterization

• Note that AOS, Data Management, and Batch are the same role in the Service Fabric. You need to size for
these three workloads combined, and not separately as you did with Microsoft Dynamics AX 2012.

- The same variability factors for SQL Server apply here.

**High availability**

- Ensure that you have at least 1 to 2 more AOS available than you estimate.
- Ensure that you have at least 3 to 4 virtual hosts available.

**Management reporter**

In most cases, unless used extensively, the recommended minimum requirements using two nodes should work well. Only in cases where there is heavy use will you need more than two nodes, after which you can scale as needed.

**SQL Server Reporting Services**

For the current release of Finance + Operations, only one SSRS node can be deployed. Monitor your SSRS node while testing and increase the number of cores available for SSRS as needed. Make sure that you have a preconfigured secondary node available on a virtual host that is different than the SSRS VM. This is important if there is an issue with the virtual machine that hosts SSRS or the virtual host. If this the case, the node would need to be replaced.

**Environment Orchestrator**

The Orchestrator service is the service that manages your deployment and the related communication with LCS. This service is deployed as the primary Service Fabric service and requires at least three VMs. This service is co-located with the Service Fabric orchestration services. This should be sized to the peak load of the cluster. For more information, see Plan and prepare your Service Fabric Standalone cluster deployment.

**Virtualization and oversubscription**

Mission critical services like the AOS should be hosted on Virtual hosts that have dedicated resources – core, memory, and disk.

**Authentication methods**

The following authentication methods are used with on-premises deployments:

- **Azure Active Directory (Azure AD)** - Azure AD is the authentication method used to log in to LCS. Azure AD is used configure the LCS Local Agent. For more information, see What is Azure Active Directory?

- **Active Directory Domain Services (AD DS)** - The machines that host Finance + Operations components must belong to an Active Directory domain. You must configure Active Directory Domain Services (AD DS) in native mode. For more information, see Active Directory Domain Services.

- **Active Directory Federation Services (AD FS)** - AD FS is the authentication method used in an on-premises deployment. AD FS provides access control and single sign on across a wide variety of applications including Microsoft 365, cloud-based SaaS applications, and applications on the corporate network.
  - For the IT organization, it enables you to provide sign on and access control to both modern and legacy applications, on-premises and in the cloud, based on the same set of credentials and policies.
  - For the user, it provides seamless sign on using the same, familiar account credentials.
  - For the developer, it provides an easy way to authenticate users whose identities live in the organizational directory. This means you can focus your efforts on your application, not authentication or identity.

  For more information, see Active Directory Federation Services.

**Data stored in Azure data centers**
The on-premises deployment option for Finance + Operations stores core customer data on-premises. Core customer data is a subset of the customer data definition provided in the Microsoft Trust Center.

The following table outlines the services that are used to store customer data in Azure data centers located in the United States. Services include Lifecycle Services (LCS), Microsoft Office signup portal, and Azure Active Directory. These services enable initial onboarding, initiation, tracking of support incidents, and service updates and upgrades. All other customer data, referred to as core customer data, is stored on-premises.

<table>
<thead>
<tr>
<th>SUPPORTING SERVICES</th>
<th>CUSTOMER DATA DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Dynamics Lifecycle Services</td>
<td>Project content and files are stored in a project. This includes application configuration data, code, metadata, and data assets that include the application and business process models.</td>
</tr>
<tr>
<td>Microsoft Office signup portal</td>
<td>Customer information that is collected during the onboarding process.</td>
</tr>
<tr>
<td>Microsoft Azure Active Directory</td>
<td>Authentication for LCS and Azure DevOps.</td>
</tr>
</tbody>
</table>

Additional services or components can be configured to extend an on-premises deployment as needed; however, configuration choices may cause core customer data to be transferred outside of the customer’s data center. For example, configuring data management features that are used to integrate external services with an on-premises deployment may result in the transfer of core customer data outside the on-premises deployment.

Next steps

After you’ve completed the planning activities mentioned in this topic, you can begin the procedures listed in the Onboard section of the On-premises deployment home page.

Be sure to refer to the On-premises deployment home page throughout your implementation for more information about planning, deployment, maintenance, and troubleshooting.
This topic explains which versions of dependent software are compatible with different versions of Microsoft Dynamics 365 Finance + Operations (on-premises).

**Microsoft Windows Server**

Both Microsoft Windows Server Standard and Microsoft Windows Server Datacenter are supported.

<table>
<thead>
<tr>
<th>VERSION</th>
<th>SUPPORTED SINCE</th>
<th>END OF LIFE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Windows Server 2019</td>
<td>10.0.17</td>
<td>Not available</td>
</tr>
<tr>
<td>Microsoft Windows Server 2016</td>
<td>Original release</td>
<td>10.0.26</td>
</tr>
</tbody>
</table>

**NOTE**

Only en-US operating system installations are supported.

**Microsoft SQL Server**

Both Microsoft SQL Server Standard Edition and Enterprise Edition are supported.

This section covers the following SQL Server components:

- Database Engine
- SQL Server Reporting Services (SSRS)
- SQL Server Integration Services (SSIS)

<table>
<thead>
<tr>
<th>VERSION</th>
<th>SUPPORTED SINCE</th>
<th>END OF LIFE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft SQL Server 2019</td>
<td>10.0.21</td>
<td>Not available</td>
</tr>
<tr>
<td>Microsoft SQL Server 2016 SP2</td>
<td>10.0.9</td>
<td>10.0.28</td>
</tr>
<tr>
<td>Microsoft SQL Server 2016 SP1</td>
<td>Original release</td>
<td>10.0.14</td>
</tr>
</tbody>
</table>

**IMPORTANT**

Using multiple versions of Microsoft SQL Server throughout a single environment is not supported.

**Active Directory Federation Services (AD FS)**

Active Directory Federation Services (AD FS) is a server role that can be installed on a machine running Windows Server.
<table>
<thead>
<tr>
<th>VERSION</th>
<th>SUPPORTED SINCE</th>
<th>END OF LIFE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Directory Federation Services on Windows Server 2019</td>
<td>10.0.17</td>
<td>Not available</td>
</tr>
<tr>
<td>Active Directory Federation Services on Windows Server 2016</td>
<td>Original release</td>
<td>10.0.26</td>
</tr>
</tbody>
</table>

**IMPORTANT**

- AD FS on Windows Server 2016 only supports authentication through the Azure Active Directory Authentication Library (ADAL).
- In order to uptake the upcoming migration to the Microsoft Authentication Library, you need to deploy your AD FS on Windows Server 2019 (MSAL). For more information, see Migrate applications to the Microsoft Authentication Library (MSAL).

### Minimum Azure Service Fabric runtime

Your Service Fabric cluster should always be on a supported version according to the official documentation, Service Fabric supported versions.

<table>
<thead>
<tr>
<th>MINIMUM VERSION</th>
<th>REQUIRED SINCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Fabric runtime 7.2</td>
<td>10.0.17</td>
</tr>
<tr>
<td>Service Fabric runtime 7.1</td>
<td>10.0.14</td>
</tr>
</tbody>
</table>

### Minimum Microsoft .NET Framework runtime

The requirements for .NET Framework are specified on a per node basis. For specific features and versions, see Set up and deploy on-premises environments (Platform update 41 and later).

<table>
<thead>
<tr>
<th>MINIMUM VERSION</th>
<th>REQUIRED SINCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft .NET Framework version 4.7.2</td>
<td>10.0.11</td>
</tr>
</tbody>
</table>

### Microsoft Office Server

Office Server is an optional component. For more information, see Configure document preview.

<table>
<thead>
<tr>
<th>VERSION</th>
<th>SUPPORTED SINCE</th>
<th>END OF LIFE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Office Server 2017</td>
<td>10.0.0</td>
<td>Not available</td>
</tr>
</tbody>
</table>
Before you begin the hardware and infrastructure sizing process for an on-premises environment, familiarize yourself with the System requirements for cloud deployments and Setup and deployment instructions to gain a solid understanding off the underlying infrastructure.

**NOTE**
Pay close attention to the system setup best practices for optimum performance.

After you have reviewed the documentation, you can start the process of estimating your transactional and concurrent user volume and sizing your environment based on the average core throughput.

**Factors that affect sizing**

All the factors shown in the following illustration contribute to sizing. The more detailed information that is collected, the more precisely you can determine sizing. Hardware sizing, without supporting data, is likely to be inaccurate. The absolute minimum requirement for necessary data is the peak transaction line load per hour.

Viewed from left to right, the first and most important factor needed to accurately estimate sizing is a transaction profile or a transaction characterization. It's important to always find the peak transactional volume per hour. If there are multiple peak periods, then these periods need to be accurately defined.

As you understand the load that impacts your infrastructure, you also need to understand more detail about these factors:

- **Transactions** – Typically transactions have certain peaks throughout the day/week. This mostly depends
on the transaction type. Time and expense entries usually show peaks once per week, whereas Sales order entries often come in bulk via integration or trickle in during the day.

- **Number of concurrent users** – The number of concurrent users is the second most important sizing factor. You cannot reliably get sizing estimates based on the number of concurrent users, so if this is the only data you have available, estimate an approximate number, and then revisit this when you have more data. An accurate concurrent user definition means that:
  - Named users are not concurrent users.
  - Concurrent users are always a subset of named users.
  - Peak workload defines the maximum concurrency for sizing.

Criteria for concurrent users is that the user meets all the following criteria:
  - Logged on.
  - Working transactions/inquiries at the time of counting.
  - Not an idle session.

- **Data composition** – This is mostly about how your system will be set up and configured. For example, how many legal entities you will have, how many items, how many BOM levels, and how complex your security setup will be. Each of those factors may have a small impact on performance, so these factors can be offset by using smart choices when it comes to infrastructure.

- **Extensions** – Customizations can be simple or complex. The number of customizations and the nature of complexity and usage has a varied impact on the size of the infrastructure needed. For complex customizations, it’s advised to conduct performance evaluations to ensure that they are not only tested for efficiency but also help understand the infrastructure needs. This is even more critical when the extensions are not coded according to best practices for performance and scalability.

- **Reporting and analytics** – These factors typically include running heavy queries against the various databases in the system. Understanding and reducing the frequency when expensive reports run will help you understand the impact of them.

- **Third-party solutions** – These solutions, like ISVs, have the same implications and recommendations as extensions.

### Sizing your environment

To understand your sizing requirements, you need to know the peak volume of transactions that you need to process. Most auxiliary systems, like Management Reporter or SSRS, are less mission critical. As a result, this document focuses mostly on AOS and SQL Server.

**NOTE**

In general, the compute tiers scale out and should be set up in an N+1 fashion, meaning if you estimate three AOS, add a fourth AOS. The database tier should be set up in an Always On highly-available setup.

### SQL Server (OLTP)

#### Sizing

- 3K to 15K transaction lines per hour per core on DB server.

- Typical AOS-to-SQL core ratio 3:1 for the primary SQL Server. Additional cores are required based on the chosen high availability configuration.
  - Processing database-heavy operations may regress this to 2:1.
The following factors influence variations:

- Parameter settings in use.
- Levels of extensions.
- Usage of additional functionality, such as database log and alerts. Extreme database logging will further reduce throughput per hour per core below 3K lines.
- Complexity of data composition – A simple chart of accounts versus a detailed fine-grained chart of accounts has implications on throughput (as an example).
- Transaction characterization.
- 2 GB to 16 GB memory for each core.
- Auxiliary databases on DB server such as Management reporter and SSRS databases.
- Temp DB = 15% of DB size, with as many files as physical processors.
- SAN size and throughput based on total concurrent transaction volume/usage.

High availability

We recommend always utilizing SQL Server in either a cluster or mirroring setup. The second SQL node should have the same number of cores as the primary node.

Active Directory Federation Services (AD FS)

For AD FS sizing, see the AD FS Server Capacity documentation.

A sizing spreadsheet is available for planning the number of instances in your deployment.

AOS (Online and batch)

Sizing

- Sizing by transaction volume/usage
  - 2K to 6K lines per core
  - 16 GB per instance
  - Standard box – 4 to 24 cores
  - 10 to 15 Enterprise users per core
  - 15 to 25 Activity users per core
  - 25 to 50 Team members per core

- Batch
  - 1 to 4 batch threads per core
  - Size based on batch window characterization

- Note that the AOS, Data Management, and Batch are on the same role in the Service Fabric. You need to size for these three workloads combined, and not separate these like in Microsoft Dynamics AX 2012.

- The same variability factors for SQL Server apply here.

High availability

- Ensure that you have at least 1 to 2 more AOS available than you estimate.
- Ensure that you have at least 3 to 4 virtual hosts available.

Management reporter

In most cases, unless used extensively, the recommended minimum requirements using two nodes should work well. Only in cases where there is heavy use will you need more than two nodes. Please scale as needed.
SQL Server Reporting Services

For the general availability release, only one SSRS node can be deployed. Monitor your SSRS node while testing and increase the number of cores available for SSRS on a need basis. Make sure that you have a preconfigured secondary node available on a virtual host that is different than the SSRS VM. This is important if there is an issue with the virtual machine that hosts SSRS or the virtual host. If this the case, they would need to be replaced.

Starting with version 10.0.17, it is possible to configure additional SSRS nodes to achieve high availability. For more information, see [Configure high availability for SQL Server Reporting Services (SSRS) nodes](#).

Environment Orchestrator

The Orchestrator service is the service that manages your deployment and the related communication with LCS. This service is deployed as the primary Service Fabric service and requires at least three VMs. This service is co-located with the Service Fabric orchestration services. This and should be sized to the peak load of the cluster. For more information, see [Plan and prepare your Service Fabric Standalone cluster deployment](#).

Virtualization and oversubscription

Mission critical services like the AOS should be hosted on Virtual hosts that have dedicated resources – cores, memory, and disk.
This topic explains authentication in Dynamics 365 Finance + Operations (on-premises). This topic also provides background information about how the process works so that if you encounter issues with authentication you can work to resolve them.

The URL for Active Directory Federation Services (AD FS)

The first part of the authentication process is to provide the URL for Active Directory Federation Services (AD FS). This URL will be similar to: https://adfs.contoso.com/adfs/.well-known/openid-configuration

You’ll find this URL in the deployment instructions found in Configure AD FS. During deployment, the URL is used to set various options in the AOS startup variables of each AOS instance. These startup variables reside in an .XML config file located in a Service Fabric directory. This directory will vary from machine to machine, but the path should look similar to:

C:\ProgramData\SF\AOS_10\Fabric\work\Applications\AXSFType_App218\AXSF.Package.1.0.xml

XML configuration file

There is a file called AXSF.Package.Current.xml. This file will be a copy of the AXSF.Package.1.0.xml in Finance and Operations deployments. The AXSF.Package.Current.xml file represents the variable that have been used to initialize the currently running AOS instance (AxService.exe).

Within this configuration file (which is on each AOS machine), you’ll find some sections that are set from the Lifecycle Services (LCS) deployment setting for AD FS.

```xml
  <Section Name="Aad">
    <Parameter Name="AADIssuerNameFormat" Value="http://ADFS.contoso.com/{0}/services/trust" />
    <Parameter Name="AADLoginWsFedEndpointFormat" Value="https://ADFS.contoso.com/{0}/wsFed" />
    <Parameter Name="AADMetadataLocationFormat" Value="https://ADFS.contoso.com/FederationMetadata/2007-06/FederationMetadata.xml" />
    <Parameter Name="AADTenantId" Value="adfs" />
    <Parameter Name="AADValidAudience" Value="https://ax.contoso.com/" />
    <Parameter Name="ACSServiceEndpoint" Value="https://accounts.accesscontrol.windows-ppe.net/tokens/OAuth/2" />
    <Parameter Name="AADTenantId" Value="adfs" />
    <Parameter Name="AADValidAudience" Value="https://ax.contoso.com/" />
    <Parameter Name="ACSServiceEndpoint" Value="https://accounts.accesscontrol.windows-ppe.net/tokens/OAuth/2" />
    <Parameter Name="ACSServicePrincipal" Value="00000001-0000-0000-c000-000000000000" />
    <Parameter Name="ADFSEndpoint" Value="https://ADFS.contoso.com/adfs/" />
    <Parameter Name="ADFSIdentifier" Value="http://ADFS.contoso.com/adfs/services/trust" />
    <Parameter Name="FederationMetadataLocation" Value="https://ADFS.contoso.com/FederationMetadata/2007-06/FederationMetadata.xml" />
    <Parameter Name="Realm" Value="spn:00000015-0000-0000-c000-000000000000" />
    <Parameter Name="TenantDomainGUID" Value="adfs" />
    <Parameter Name="TrustedServiceAppIds" Value="913c6de4-2a4a-4a61-a9ce-945d2b2ce2e0" />
  </Section>
```

You will also find the following sections.
The settings shown above represent a deployment configured with Microsoft 365 compatibility. For more information, see AD FS Microsoft 365 compatibility.

Configuration values used by the AOS

The AOS uses the configuration values above to determine where to redirect an unauthenticated request when a user sends a request to the application URL.

1. Request is sent by the browser to the application URL (https://ax.contoso.com/namespaces/AXSF/).
2. The request is processed by the Gateway and gets forwarded to an AOS node that accepts interactive sessions.
3. The request reaches the AOS server and checks for the authentication cookies.
4. No authentication is present so the AOS server returns a redirect request for the user to authenticate with AD FS. At this point, the AOS also sets an affinity cookie to bind the user session to that AOS.
5. The Gateway receives the response and forwards it back to the browser.
6. The browser receives the redirect request and displays the AD FS authentication page so the user signs in.
7. When successfully authenticated against AD FS, the AD FS then redirects the user back to the application URL and provides the authentication cookies.
8. The Gateway receives this response and forwards the affinitized request to the appropriate AOS node.
9. The AOS checks the authentication information provided and checks against the UserInfo table to determine whether the user is allowed to access the application and which permissions are available.

If values in the AOS config file are incorrect, then that typically means the value provided for the AD FS endpoint when deploying the environment was incorrect. The easiest thing is to delete and redeploy the environment from LCS with the correct value. It is possible to manually edit the configuration files, but to be safe, do a redeploy. Otherwise you will need to manually change the values after each servicing operation on each AOS node. If you do edit the config files, then you need to restart the AOS service (AxService.exe) for it to take effect. You can do that from the Service Fabric explorer (right-click the AOS node under Nodes, choose Restart, and then wait at least a minute for the status to change to green). You can also reboot the machine.

Receiving a 500 error when accessing the application URL is an indication that there may be an invalid URL for AD FS. This is because on startup the AOS will use that URL to obtain information from the AD FS server. If the URL is incorrect or inaccessible, the AOS will be unable to start.

AD FS

The second part of the authentication process is AD FS itself. On the AD FS server if you open AD FS Management ((from Control Panel > System and Security > Administrative Tools), and go to Application groups, you'll find a group called Microsoft Dynamics 365 for Operations On-premises.
Within this group, the settings for AD FS for your Dynamics 365 application are stored.

AD FS uses the client ID and the URLs to determine whether the request for access should be honored. You will notice that the client ID from the screenshot above matches the IDs specified in the OfficeApps and OpenIDConnect sections from earlier. If both the client ID and the redirect URL don't match what the AOS is requesting, then AD FS will deny the request to authenticate. If that happens, you’ll find an error in the event log on the AD FS server. There’s a special event log for AD FS under Application and Services logs > AD FS > Admin.

If any of the AD FS application group setup is incorrect, you’re likely see an error in the event log that explains the value it was looking for, so you can determine what is set incorrectly.
You must use Microsoft Dynamics Lifecycle Services (LCS) to deploy and update an instance of Dynamics 365 Finance + Operations (on-premises). After you purchase a server and user license through the Volume Licensing flow or the Dynamics Price List flow, see the topic, Buy Finance + Operations (on-premises), to create an Azure AD account or use an existing Azure AD account, and then complete all the sign-up steps. You will be redirected to LCS, where an on-premises implementation project will be provisioned for you.

The on-premises project has all the tools that you require in order to implement, maintain, and operate an on-premises solution. Here are some of the tools that are available in the on-premises project:

- **Methodology** – The on-premises methodology provides best practices that will help customers implement and manage on-premises projects.
- **Business process modeler** – Business process modeler (BPM) is used to capture requirements and do fit gap analysis.
- **Cloud-hosted environments** – Cloud-hosted environments are used to deploy developer and build topologies, and to complete Dev Application Lifecycle Management (ALM) for on-premises solutions.
- **Code upgrade** – These tools will help you upgrade code to a newer release.
- **Issue search** – Search for published KBs that are related to application and platform issues.
- **Localization and translation** – Localize and translate assets.
- **Support** – File and track support incidents.
- **Project users** – Assign users to a project.
- **Project settings** – Edit project-level settings, such as connectors, the project name, organization users, and the license number.
- **Asset library** – The Asset library is a library for various assets, such as packages.
- **SharePoint online library** – Connect to an online Microsoft SharePoint library.

To start your on-premises implementation, you must follow the steps in the methodology to correctly set up the project, deploy the developer and build environments, and then deploy sandbox and production environments. To help you with deployments, two environment slots are pre-allocated to the on-premises project. One slot is for a sandbox environment, and the other slot is for a production environment. These slots will be used during the Servicing flow to help guarantee that packages are tested in the sandbox environment before they are applied in the production environment.
This topic explains how to plan, set up, and deploy Microsoft Dynamics 365 Finance + Operations (on-premises) with Platform update 41 and later. Platform update 41 is available with version 10.0.17.

The Local Business Data Yammer group is available. There, you can post any questions or feedback that you have about the on-premises deployment.

**Finance + Operations components**

The Finance + Operations application consists of three main components:

- Application Object Server (AOS)
- Business Intelligence (BI)
- Financial Reporting/Management Reporter

These components depend on the following system software:

- Microsoft Windows Server (Only English-language operating system installations are supported.)
- Microsoft SQL Server

**IMPORTANT**

Full-Text Search must be enabled.

- SQL Server Reporting Services (SSRS)
  
  SSRS is deployed on BI virtual machines (VMs). The SSRS nodes should also have a Database Engine instance that is running locally.

- SQL Server Integration Services (SSIS)
  
  SSIS is deployed on AOS VMs.

- SQL Server Management Studio

- Standalone Microsoft Azure Service Fabric

- Microsoft Windows PowerShell 5.0 or later

- Active Directory Federation Services (AD FS) on Windows Server

- Domain controller
The domain controller must be Microsoft Windows Server 2012 R2 or later, and it must have a domain functional level of 2012 R2 or more. For more information about domain functional levels, see the following topics:

- What Are Active Directory Functional Levels?
- Understanding Active Directory Domain Services (AD DS) Functional Levels
- Full 2-way trust

• Optional but highly recommended: Active Directory Certificate Services (AD CS) on Windows Server

For supported versions, see Microsoft Dynamics 365 Finance + Operations (on-premises) supported software.

LCS

Finance + Operations bits are distributed through Microsoft Dynamics Lifecycle Services (LCS). Before you can deploy, you must purchase license keys through the Enterprise Agreements channel and set up an on-premises project in LCS. Deployments can be initiated only through LCS. For more information about how to set up on-premises projects in LCS, see Set up on-premises projects in Lifecycle Services (LCS).

Authentication

The on-premises application works with AD FS. To interact with LCS, you must also configure Azure Active Directory (Azure AD). To complete the deployment and configure the LCS local agent, you must have Azure AD. If you don’t already have an Azure AD tenant, you can get one for free by using one of the options that Azure AD provides. For more information, see Quickstart: Set up a tenant.

Standalone Service Fabric

Finance + Operations uses standalone Service Fabric. For more information, see the Service Fabric documentation.

Setup of Finance + Operations will deploy a set of applications inside Service Fabric. During deployment, each node in the cluster will be defined through configuration so that it has one of the following node types:

- **AOSNodeType** – Nodes of this type host AOS (business logic).
- **OrchestratorType** – Nodes of this node type work as Service Fabric Primary nodes, and host deployment and servicing logic.
- **ReportServerType** – Nodes of this type host SSRS and reporting logic.
- **MRType** – Nodes of this type host Management Reporter logic.

Infrastructure

Finance + Operations falls under the standard Microsoft support policy about operation on non-Microsoft virtualization platforms, specifically VMware. For more information, see Support policy for Microsoft software that runs on non-Microsoft hardware virtualization software. In short, Microsoft supports its products in this environment. However, if Microsoft is asked to investigate an issue, we might first ask the customer to reproduce the issue without the virtualization platform or on the Microsoft virtualization platform.

If you're using VMware, you must implement the fixes that are documented on the following webpages:

- After upgrading a virtual machine to hardware version 11, network dependent workloads experience
performance degradation (2129176)

- Several issues with vmxnet3 virtual adapter

### WARNING

Dynamics 365 Finance + Operations (on-premises) is not supported on any public cloud infrastructure, including Microsoft Azure Cloud services. However, it is supported to run on Microsoft Azure Stack Hub.

The hardware configuration includes the following components:

- A standalone Service Fabric cluster that is based on Windows Server VMs
- SQL Server (Both Clustered SQL and Always-On are supported.)
- AD FS for authentication
- Server Message Block (SMB) version 3 file share for storage
- Optional: Microsoft Office Server

For more information, see [System requirements for on-premises deployments](#).

#### IMPORTANT

For supported versions, see [Microsoft Dynamics 365 Finance + Operations (on-premises) supported software](#).

### Hardware layout

Plan your infrastructure and Service Fabric cluster, based on the recommended sizing in [Hardware sizing requirements for on-premises environments](#). For more information about how to plan the Service Fabric cluster, see [Plan and prepare your Service Fabric standalone cluster deployment](#).

The following table shows an example of a hardware layout. This example is used throughout this topic to demonstrate the setup. When you complete the setup, you will have to replace the machine names and IP addresses that are provided in the following instructions with the names and IP addresses for the machines in your environment.

#### NOTE

The Primary node of the Service Fabric cluster must have at least three nodes. In this example, OrchestratorType is designated as the Primary node type. If you have a node type that has more than three VMs, consider making that node type your Primary (Seed) node type to help increase the reliability of the cluster.

<table>
<thead>
<tr>
<th>MACHINE PURPOSE</th>
<th>SERVICE FABRIC NODE TYPE</th>
<th>MACHINE NAME</th>
<th>IP ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain controller</td>
<td></td>
<td>DAX7SQLAOADC1</td>
<td>10.179.108.2</td>
</tr>
<tr>
<td>AD FS</td>
<td></td>
<td>DAX7SQLAOADFS1</td>
<td>10.179.108.3</td>
</tr>
<tr>
<td>File server</td>
<td></td>
<td>DAX7SQLAOFILE1</td>
<td>10.179.108.4</td>
</tr>
<tr>
<td>SQL Always-On cluster</td>
<td></td>
<td>DAX7SQLAOSQLA01</td>
<td>10.179.108.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DAX7SQLAOSQLA02</td>
<td>10.179.108.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DAX7SQLAOSQLA</td>
<td>10.179.108.9</td>
</tr>
</tbody>
</table>
The following table shows an example of a hardware layout where batch execution and interactive sessions are run in dedicated nodes. For more information, see Configure batch-only and interactive-only AOS nodes in on-premises deployments.

<table>
<thead>
<tr>
<th>MACHINE PURPOSE</th>
<th>SERVICE FABRIC NODE TYPE</th>
<th>MACHINE NAME</th>
<th>IP ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td></td>
<td>SQLAOCLIENT1</td>
<td>10.179.108.11</td>
</tr>
<tr>
<td>AOS 1</td>
<td>AOSNodeType</td>
<td>SQLAOSF1AOS1</td>
<td>10.179.108.12</td>
</tr>
<tr>
<td>AOS 2</td>
<td>AOSNodeType</td>
<td>SQLAOSF1AOS2</td>
<td>10.179.108.13</td>
</tr>
<tr>
<td>AOS 3</td>
<td>AOSNodeType</td>
<td>SQLAOSF1AOS3</td>
<td>10.179.108.14</td>
</tr>
<tr>
<td>Orchestrator 1</td>
<td>OrchestratorType</td>
<td>SQLAOSF1ORCH1</td>
<td>10.179.108.21</td>
</tr>
<tr>
<td>Orchestrator 2</td>
<td>OrchestratorType</td>
<td>SQLAOSF1ORCH2</td>
<td>10.179.108.22</td>
</tr>
<tr>
<td>Orchestrator 3</td>
<td>OrchestratorType</td>
<td>SQLAOSF1ORCH3</td>
<td>10.179.108.23</td>
</tr>
<tr>
<td>Management Reporter node</td>
<td>MRType</td>
<td>SQLAOSMR1</td>
<td>10.179.108.31</td>
</tr>
<tr>
<td>SSRS node 1</td>
<td>ReportServerType</td>
<td>SQLAOSFBI1</td>
<td>10.179.108.41</td>
</tr>
<tr>
<td>Domain controller</td>
<td></td>
<td>DAX7SQLAODC1</td>
<td>10.179.108.2</td>
</tr>
<tr>
<td>AD FS</td>
<td></td>
<td>DAX7SQLAOADFS1</td>
<td>10.179.108.3</td>
</tr>
<tr>
<td>File server</td>
<td></td>
<td>DAX7SQLAOFILE1</td>
<td>10.179.108.4</td>
</tr>
<tr>
<td>SQL Always-On cluster</td>
<td></td>
<td>DAX7SQLAOSQLA01</td>
<td>10.179.108.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DAX7SQLAOSQLA02</td>
<td>10.179.108.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DAX7SQLAOSQLA</td>
<td>10.179.108.9</td>
</tr>
<tr>
<td>Client</td>
<td></td>
<td>SQLAOCLIENT1</td>
<td>10.179.108.11</td>
</tr>
<tr>
<td>AOS 1</td>
<td>BatchOnlyAOSNodeType</td>
<td>SQLAOSF1AOS1</td>
<td>10.179.108.12</td>
</tr>
<tr>
<td>AOS 2</td>
<td>BatchOnlyAOSNodeType</td>
<td>SQLAOSF1AOS2</td>
<td>10.179.108.13</td>
</tr>
<tr>
<td>AOS 3</td>
<td>BatchOnlyAOSNodeType</td>
<td>SQLAOSF1AOS3</td>
<td>10.179.108.14</td>
</tr>
<tr>
<td>AOS 4</td>
<td>InteractiveOnlyAOSNodeType</td>
<td>SQLAOSF1AOS4</td>
<td>10.179.108.15</td>
</tr>
</tbody>
</table>
### Overview of the setup process

You must complete the following steps to set up the infrastructure for Finance + Operations. By reading all the steps before you begin, you can more easily plan your setup.

1. Plan your domain name and DNS zones
2. Plan and acquire your certificates
3. Plan your users and service accounts
4. Create DNS zones, and add A records
5. Join VMs to the domain
6. Download setup scripts from LCS
7. Describe your configuration
8. Configure certificates
9. Set up SSIS
10. Set up SSRS
11. Set up VMs
12. Set up a standalone Service Fabric cluster
13. Configure LCS connectivity for the tenant
14. Set up file storage
15. Set up SQL Server
16. Configure the databases
17. Encrypt credentials
18. Configure AD FS
19. Configure a connector, and install an on-premises local agent
20. Tear down CredSSP, if remoting was used
21. Deploy your Finance + Operations environment from LCS
22. Connect to your Finance + Operations environment

---

**Setup**

**Prerequisites**
Before you start the setup, the following prerequisites must be in place. The setup of these prerequisites is out of the scope of this document.

- Active Directory Domain Services (AD DS) must be installed and configured in your network.
- AD FS must be deployed.
- SQL Server must be installed on the SSRS machines.
- SSRS must be installed (but not configured) in **Native** mode on the SSRS machines.
- Optional: AD CS is installed and configured in your network.

The following table shows the prerequisite software that is installed on the VMs by the infrastructure setup scripts that are downloaded from LCS.

<table>
<thead>
<tr>
<th>NODE TYPE</th>
<th>COMPONENT</th>
<th>DETAILS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AOS</td>
<td>SNAC – ODBC driver 13</td>
<td>ODBC driver 13.1</td>
</tr>
<tr>
<td>AOS</td>
<td>SNAC – ODBC driver 17.5.x</td>
<td>ODBC driver 17.5.2</td>
</tr>
<tr>
<td>AOS</td>
<td>The Microsoft .NET Framework version 2.0–3.5 (CLR 2.0)</td>
<td><strong>Windows features</strong>: .NET-Framework-Features, .NET-Framework-Core, .NET-HTTP-Activation, .NET-Non-HTTP-Activ</td>
</tr>
<tr>
<td>AOS</td>
<td>The Microsoft .NET Framework version 4.0–4.6 (CLR 4.0)</td>
<td><strong>Windows features</strong>: .NET-Framework-45-Features, .NET-Framework-45-Core, .NET-Framework-45-ASPNET, .NET-WCF-Services45, .NET-WCF-TCP-PortSharing45</td>
</tr>
<tr>
<td>AOS</td>
<td>The Microsoft .NET Framework version 4.7.2 (CLR 4.0)</td>
<td><a href="https://dotnet.microsoft.com/download/thank-you/net472-offline">https://dotnet.microsoft.com/download/thank-you/net472-offline</a></td>
</tr>
<tr>
<td>AOS</td>
<td>SQL Server Management Studio 17.9.1</td>
<td>SSMS 17.9.1</td>
</tr>
<tr>
<td>AOS</td>
<td>Microsoft Visual C++ Redistributable Packages for Microsoft Visual Studio 2017</td>
<td>Go to <a href="https://lcs.dynamics.com/V2/SharedAssetLibrary">https://lcs.dynamics.com/V2/SharedAssetLibrary</a>, select Model as the asset type, and then select VC++ 17 Redistributables.</td>
</tr>
<tr>
<td>BI</td>
<td>The .NET Framework version 2.0–3.5 (CLR 2.0)</td>
<td><strong>Windows features</strong>: .NET-Framework-Features, .NET-Framework-Core, .NET-HTTP-Activation, .NET-Non-HTTP-Activ</td>
</tr>
<tr>
<td>NODE TYPE</td>
<td>COMPONENT</td>
<td>DETAILS</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>BI</td>
<td>The .NET Framework version 4.0–4.6 (CLR 4.0)</td>
<td>Windows features: NET-Framework-45-Features, NET-Framework-45-Core, NET-Framework-45-ASPNET, NET-WCF-Services45, NET-WCF-TCP-PortSharing45</td>
</tr>
<tr>
<td>BI</td>
<td>The .NET Framework version 4.7.2 (CLR 4.0)</td>
<td><a href="https://dotnet.microsoft.com/download/thank-you/net472-offline">https://dotnet.microsoft.com/download/thank-you/net472-offline</a></td>
</tr>
<tr>
<td>BI</td>
<td>SQL Server Management Studio 17.9.1</td>
<td>SSMS 17.9.1</td>
</tr>
<tr>
<td>MR</td>
<td>The .NET Framework version 4.0–4.6 (CLR 4.0)</td>
<td>Windows features: NET-Framework-45-Features, NET-Framework-45-Core, NET-Framework-45-ASPNET, NET-WCF-Services45, NET-WCF-TCP-PortSharing45</td>
</tr>
<tr>
<td>MR</td>
<td>The .NET Framework version 4.7.2 (CLR 4.0)</td>
<td><a href="https://dotnet.microsoft.com/download/thank-you/net472-offline">https://dotnet.microsoft.com/download/thank-you/net472-offline</a></td>
</tr>
<tr>
<td>ORCH</td>
<td>The Microsoft .NET Framework version 4.0–4.8 (CLR 4.0)</td>
<td><a href="https://dotnet.microsoft.com/download/thank-you/net48-offline">https://dotnet.microsoft.com/download/thank-you/net48-offline</a></td>
</tr>
</tbody>
</table>

**Step 1. Plan your domain name and DNS zones**

We recommend that you use a publicly registered domain name for your production installation of AOS. In that way, the installation can be accessed outside the network, if outside access is required.

For example, if your company’s domain is contoso.com, your zone for Finance + Operations might be d365ffo.onprem.contoso.com, and the host names might be as follows:

- ax.d365ffo.onprem.contoso.com for AOS machines
- sf.d365ffo.onprem.contoso.com for the Service Fabric cluster

**Step 2. Plan and acquire your certificates**

Secure Sockets Layer (SSL) certificates are required to secure a Service Fabric cluster and all the applications that are deployed. For your production and sandbox workloads, we recommend that you acquire certificates from a certificate authority (CA) such as DigiCert, Comodo, Symantec, GoDaddy, or GlobalSign. If your domain is set up with AD CS, you can use the Microsoft setup scripts to create the templates and certificates. Each certificate must contain a private key that was created for key exchange, and it must be exportable to a Personal Information Exchange (.pfx) file.

Self-signed certificates can be used only for testing purposes. For the sake of convenience, the setup scripts that are provided in LCS include scripts that generate and export self-signed certificates. If you’re using self-signed scripts, you will be instructed to run the creation scripts during later steps in this topic. As has been mentioned, these certificates can be used only for testing purposes.
IMPORTANT
Microsoft plans to discontinue support for the generation of self-signed certificates through the setup scripts, in favor of automatic certificate creation through AD CS.

Here are the recommended settings for certificates:

- **Signature algorithm**: sha256RSA
- **Signature hash algorithm**: sha256
- **Public key**: RSA (2048 bits)
- **Thumbprint algorithm**: sha1

<table>
<thead>
<tr>
<th>PURPOSE</th>
<th>EXPLANATION</th>
<th>ADDITIONAL REQUIREMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Server SSL certificate</td>
<td>This certificate is used to encrypt data that is transmitted across a network between an instance of SQL Server and a client application.</td>
<td>The domain name of the certificate should match the fully qualified domain name (FQDN) of the SQL Server instance or listener. For example, if the SQL listener is hosted on machine DAX7SQLAOSQLA, the certificate's Domain Name System (DNS) name is DAX7SQLAOSQLA.contoso.com.</td>
</tr>
</tbody>
</table>
|                              |                                                                            | - **Common name (CN)**: DAX7SQLAOSQLA.contoso.com  
|                              |                                                                            | - **DNS name**: DAX7SQLAOSQLA.contoso.com  |
| Service Fabric Server certificate | This certificate is used to help secure the node-to-node communication between the Service Fabric nodes. It's also used as the server certificate that is presented to the client that connects to the cluster. | For this certificate, you can also use the wildcard SSL certificate for your domain, such as *.contoso.com. (For more information, see the text that follows this table.) Otherwise, use the following values:  |
|                              |                                                                            | - **CN**: sf.d365ffo.onprem.contoso.com  
|                              |                                                                            | - **DNS name**: sf.d365ffo.onprem.contoso.com  |
| Service Fabric Client certificate | Clients use this certificate to view and manage the Service Fabric cluster. |  |
|                              |                                                                            | - **CN**: client.d365ffo.onprem.contoso.com  
<p>|                              |                                                                            | - <strong>DNS name</strong>: client.d365ffo.onprem.contoso.com  |</p>
<table>
<thead>
<tr>
<th>PURPOSE</th>
<th>EXPLANATION</th>
<th>ADDITIONAL REQUIREMENTS</th>
</tr>
</thead>
</table>
| Encipherment certificate     | This certificate is used to encrypt sensitive information such as the SQL Server password and user account passwords. | The certificate must be created by using the Microsoft Enhanced Cryptographic Provider v1.0 provider. The certificate key usage must include Data Encipherment (10), and should not include server authentication or client authentication. For more information, see Managing secrets in Service Fabric applications.  
- **CN:** axdataenciphermentcert  
- **DNS name:** axdataenciphermentcert |
| AOS SSL certificate          | This certificate is used as the server certificate that is presented to the client for the AOS website. It's also used to enable Windows Communication Foundation (WCF)/Simple Object Access Protocol (SOAP) certificates. | You can use the same wildcard SSL certificate that you used as the Service Fabric server certificate. Otherwise, use the following values:  
- **CN:** ax.d365ffo.onprem.contoso.com  
- **DNS name:** ax.d365ffo.onprem.contoso.com |
| Session Authentication certificate | AOS uses this certificate to help secure a user's session information. | This certificate is also the File Share certificate that will be used at the time of deployment from LCS.  
- **CN:** SessionAuthentication  
- **DNS name:** SessionAuthentication |
| Data Encryption certificate  | AOS uses this certificate to encrypt sensitive information.                 | This certificate must be created by using the Microsoft Enhanced RSA and AES Cryptographic Provider provider.  
- **CN:** DataEncryption  
- **DNS name:** DataEncryption |
| Data Signing certificate     | AOS uses this certificate to encrypt sensitive information.                 | This certificate is separate from the Data Encryption certificate and must be created by using the Microsoft Enhanced RSA and AES Cryptographic Provider provider.  
- **CN:** DataSigning  
- **DNS name:** DataSigning |
<table>
<thead>
<tr>
<th>PURPOSE</th>
<th>EXPLANATION</th>
<th>ADDITIONAL REQUIREMENTS</th>
</tr>
</thead>
</table>
| Financial Reporting Client certificate | This certificate is used to help secure the communication between the Financial Reporting services and AOS. | • CN: FinancialReporting  
• DNS name: FinancialReporting |
| Reporting certificate | This certificate is used to help secure the communication between SSRS and AOS. | **Important:** Do not reuse the Financial Reporting Client certificate.  
• CN: ReportingService  
• DNS name: ReportingService |
| SSRS Web Server certificate | This certificate is used as the server certificate that is presented to the client (AOS) for the SSRS web server. | The domain name of the certificate should match the FQDN of the SSRS node.  
• CN: BI1.contoso.com  
• DNS name: BI1.contoso.com |
| On-Premises local agent certificate | This certificate is used to help secure the communication between a local agent that is hosted on-premises and on LCS. It enables the local agent to act on behalf of your Azure AD tenant, and to communicate with LCS to orchestrate and monitor deployments. | • CN: OnPremLocalAgent  
• DNS name: OnPremLocalAgent |

You can use the wildcard SSL certificate for your domain to combine the Service Fabric Server certificate and the AOS SSL certificate.

Here is an example of a Service Fabric Server certificate that is combined with an AOS SSL certificate.

**Subject name**

```
CN = *.d365ffo.onprem.contoso.com
```

**Subject alternative names**

```
DNS Name=ax.d365ffo.onprem.contoso.com  
DNS Name=sf.d365ffo.onprem.contoso.com  
DNS Name=*.d365ffo.onprem.contoso.com
```

**IMPORTANT**

You can use the wildcard certificate to help secure only the first-level subdomain of the domain that it’s issued to. Therefore, a certificate for *.onprem.contoso.com won’t be valid for ax.d365ffo.onprem.contoso.com.

**Step 3. Plan your users and service accounts**

You must create several user or service accounts for Finance + Operations to work. You must create a
A combination of group managed service accounts (gMSAs), domain accounts, and SQL accounts. The following table shows the user accounts, their purpose, and example names that will be used in this topic.

<table>
<thead>
<tr>
<th>USER ACCOUNT</th>
<th>TYPE</th>
<th>PURPOSE</th>
<th>USER NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Financial Reporting Application Service Account</td>
<td>gMSA</td>
<td>You should create this user for future proofing. Microsoft plans to enable AOS to work with the gMSA in upcoming releases. By creating this user at the time of setup, you help to ensure a seamless transition to the gMSA.*</td>
<td>Contoso\svc-FRAS$</td>
</tr>
<tr>
<td>Financial Reporting Process Service Account</td>
<td>gMSA</td>
<td></td>
<td>Contoso\svc-FRPS$</td>
</tr>
<tr>
<td>Financial Reporting Click Once Designer Service Account</td>
<td>gMSA</td>
<td></td>
<td>Contoso\svc-FRCO$</td>
</tr>
<tr>
<td>AOS Service Account</td>
<td>gMSA</td>
<td></td>
<td>Contoso\svc-AXSF$</td>
</tr>
<tr>
<td>SSRS bootstrapper Service Account</td>
<td>gMSA</td>
<td>The reporting service bootstrapper uses this account to configure the SSRS service.</td>
<td>Contoso\svc-ReportSvc$</td>
</tr>
<tr>
<td>AOS Service Account</td>
<td>Domain account</td>
<td>AOS uses this user in the general availability (GA) release.*</td>
<td>Contoso\AXServiceUser</td>
</tr>
<tr>
<td>AOS SQL DB Admin user</td>
<td>SQL user</td>
<td>Finance + Operations uses this user to authenticate with SQL**. This user will also be replaced by the gMSA user in upcoming releases***.</td>
<td>AXDBAdmin</td>
</tr>
<tr>
<td>Local Deployment Agent Service Account</td>
<td>gMSA</td>
<td>The local agent uses this account to orchestrate the deployment on various nodes.</td>
<td>Contoso\Svc-LocalAgent$</td>
</tr>
</tbody>
</table>

* These accounts should not have their regional settings changed. They should have the default EN-US region settings.

** If the password of the SQL user contains special characters, you might encounter issues during deployment.

*** The SQL user name and password for SQL authentication are secured because they are encrypted and stored in the file share.

**Step 4. Create DNS zones and add A records**

DNS is integrated with AD DS, and lets you organize, manage, and find resources in a network. The following procedures show how to create a DNS forward lookup zone and A records for the AOS host name and Service Fabric cluster. In this example, the DNS zone name is d365ffo.onprem.contoso.com, and the A records/host names are as follows:
Add a DNS zone
1. Sign in to the domain controller machine, select Start. Then open DNS Manager by entering `dnsmgmt.msc` and selecting the `dnsmgmt (DNS)` application.
2. Right-click the domain controller name in the console tree, and then select `New Zone` > `Next`.
3. Select `Primary Zone`.
4. Leave the `Store the zone in Active Directory` (available only if the DNS Server is a writeable domain controller) checkbox selected, and then select `Next`.
5. Select To all DNS Servers running on Domain Controllers in this domain: Contoso.com, and then select `Next`.
6. Select `Forward Lookup Zone`, and then select `Next`.
7. Enter the zone name for your setup, and then select `Next`. For example, enter `d365ffo.onprem.contoso.com`.
8. Select `Do not allow dynamic updates`, and then select `Next`.

Set up an A record for AOS
In the new DNS zone, for each Service Fabric cluster node of the `AOSNodeType` type, create one A record that is named `ax.d365ffo.onprem.contoso.com`. Don't create A records for the other node types.
1. Find the newly created zone under the `Forward Lookup Zones` folder in DNS Manager.
2. Select and hold (or right-click) the new zone, and then select `New Host`.
3. Enter the name and IP address of the Service Fabric node. (For example, enter `ax` as the name and `10.179.108.12` as the IP address.) Then select `Add Host`.
4. Leave both checkboxes cleared.
5. Repeat steps 1 through 4 for each additional AOS node.

Set up an A record for the orchestrator
In the new DNS zone, for each Service Fabric cluster node of the `OrchestratorType` type, create an A record that is named `sf.d365ffo.onprem.contoso.com`. Don't create A records for the other node types.
1. Select and hold (or right-click) the new zone, and then select `New Host`.
2. Enter the name and IP address of the Service Fabric node. (For example, enter `sf` as the name and `10.179.108.15` as the IP address.) Then select `Add Host`.
3. Leave both checkboxes cleared.
4. Repeat steps 1 through 3 for each additional orchestrator node.

Step 5. Join VMs to the domain
Join each VM to the domain by completing the steps in Join a Computer to a Domain. Alternatively, use the following Windows PowerShell script.

```
$domainName = Read-Host -Prompt 'Specify domain name (ex: contoso.com)'
Add-Computer -DomainName $domainName -Credential (Get-Credential -Message 'Enter domain credential')
```

**IMPORTANT**
You must restart the VMs after you join them to the domain.

Step 6. Download setup scripts from LCS
Microsoft has provided several scripts to help improve the setup experience. Follow these steps to download the
1. Sign in to LCS.
2. On the dashboard, select the Shared asset library tile.
3. Select Model as the asset type, and then, in the grid, select the row for Dynamics 365 for Operations on-premises - Deployment scripts.
4. Select Versions, and download the latest version of the zip file for the scripts.
5. After the zip file is downloaded, select and hold (or right-click) it, and then select Properties. In the Properties dialog box, select the Unblock checkbox.
6. Copy the zip file to the machine that will be used to run the scripts.
7. Unzip the files into a folder that is named infrastructure.

**Step 7. Describe your configuration**

The infrastructure setup scripts use the following configuration files to drive the setup:

- infrastructure\ConfigTemplate.xml
- infrastructure\D365FO-OP\NodeTopologyDefinition.xml
- infrastructure\D365FO-OP\DatabaseTopologyDefinition.xml

**IMPORTANT**

Make sure that all edits are made to the ConfigTemplate.xml file in this folder.

The infrastructure\ConfigTemplate.xml configuration file describes the following details:

- The service accounts that are required for the application to work
- The certificates that are required to help secure communications
- The database configuration
- The Service Fabric cluster configuration

**IMPORTANT**

When you configure the Service Fabric cluster, make sure that there are three fault domains for the Primary node type (OrchestratorType). Also make sure that no more than one type of node is deployed on a single machine.

For each Service Fabric node type, the infrastructure\D365FO-OP\NodeTopologyDefinition.xml configuration file describes the following details:

- The mapping between each node type and the application, domain and service accounts, and certificates
- Whether User Account Control (UAC) is enabled
- The prerequisites for Windows features and system software
• Whether strong name validation should be enabled
• The list of firewall ports that should be opened
• Which permissions an account requires for a machine

For each database, the infrastructure\D365FO-OP\DatabaseTopologyDefinition.xml configuration file describes the following details:

• The database settings
• The mappings between users and roles

Create gMSA and domain user accounts
1. Go to the machine that has the unzipped infrastructure scripts in the infrastructure folder.
2. Copy the infrastructure folder to the domain controller machine.
3. Open Windows PowerShell in elevated mode, change the directory to the infrastructure folder, and run the following commands.

   IMPORTANT
   These commands don't create an AxServiceUser domain user for you. You must create it yourself.

   Import-Module .\D365FO-OP\D365FO-OP.psd1
   New-D365FOGMSAAccounts -ConfigurationFilePath .\ConfigTemplate.xml
   Update-D365FOGMSAAccounts -ConfigurationFilePath .\ConfigTemplate.xml

4. If you must make changes to accounts or machines, update the ConfigTemplate.xml file in the original infrastructure folder, copy it to this machine, and then run the following command.

   Update-D365FOGMSAAccounts -ConfigurationFilePath .\ConfigTemplate.xml

Step 8. Configure certificates
1. Go to the machine that you originally unzipped the infrastructure folder to.
2. Generate certificates:
   a. If you must generate certificates, run the following commands. These commands create the certificate templates in AD CS, generate the certificates from the templates, put the certificates in the CurrentUser\My certificate store on the machine, and update the thumbprints in the XML file.

      # If you must create self-signed certs, set the generateSelfSignedCert attribute to true.
      #.\New-SelfSignedCertificates.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
      .\New-ADCSCertificates.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -CreateTemplates
      .\New-ADCSCertificates.ps1 -ConfigurationFilePath .\ConfigTemplate.xml

      NOTE
      You must run these commands on a domain controller machine, or on a machine that is running Windows Server and that has Remote Server Administration Tools (RSAT) installed.

   b. If you must reuse any certificates and therefore don’t have to generate certificates for them, set the generateADCSCert tag to false.
3. If you're using SSL certificates that were previously generated, skip certificate generation, update the thumbprints in the ConfigTemplate.xml file. The certificates must be installed in theCurrentUser\My certificate store, and their private keys must be exportable.

**WARNING**

Because of a leading non-printable special character, the presence of which is difficult to determine, the Certificate Manager tool (certlm.msc) should not be used to copy thumbprints. If the non-printable special character is present, you will receive the following error message: "X509 certificate not valid." To retrieve the thumbprints, see the results from Windows PowerShell commands, or run the following commands in Windows PowerShell.

```powershell
dir cert:\CurrentUser\My
dir cert:\LocalMachine\My
dir cert:\LocalMachine\Root
```

4. In the ProtectTo tag for each certificate, specify a semicolon-separated list of Active Directory users or groups. Only users and groups that are specified in the ProtectTo tag will have the permissions to import the certificates that are exported by using the scripts. The scripts don't support passwords to help protect the exported certificates.

5. Export the certificates into .pfx files. As part of the export process, the following command will check that the correct cryptographic provider is set for your certificates.

```powershell
# Exports .pfx files into a directory VMs\<VMName>. All the certs will be written to the infrastructure\Certs folder.
.
\Export-PfxFiles.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

**Step 9. Set up SSIS**

To enable Data management and SSIS workloads, you must install SSIS on each AOS VM. Follow these steps on each AOS VM.

1. Verify that the machine has access to the SSIS installation, and open the SSIS Setup wizard.
2. On the Feature Selection page, in the Features pane, select the Integration Services and SQL Client Connectivity SDK checkboxes.
3. Complete the setup, and verify that the installation was successful.

For more information, see Install Integration Services (SSIS).

**Step 10. Set up SSRS**

You can configure more than one SSRS node. For more information, see Configuring High Availability for SSRS nodes.

1. Before you begin, make sure that the prerequisites that are listed at the beginning of this topic are in place.

**IMPORTANT**

- You must install the Database Engine when you install SSRS.
- Do not configure the SSRS instance. The reporting service will automatically configure everything.
- Environments that were deployed with a base topology older than Platform update 41, do not need to go through the following steps. In those environments, SSRS should be configured manually according to Configure SQL Server Reporting Services for on-premises deployments.

2. For each BI node, follow these steps:
a. Copy the infrastructure folder. Then open Windows PowerShell in elevated mode, and go to the folder.

b. Run the following commands.

```
Initialize-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName BI
Configure-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName BI
```

The Initialize-Database.ps1 script maps the gMSA to the following databases and roles.

<table>
<thead>
<tr>
<th>USER</th>
<th>DATABASE</th>
<th>DATABASE ROLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>svc-ReportSvc$</td>
<td>master</td>
<td>db_owner</td>
</tr>
<tr>
<td>svc-ReportSvc$</td>
<td>msdb</td>
<td>db_datareader, db_datawriter,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>db_securityadmin</td>
</tr>
</tbody>
</table>

The Configure-Database.ps1 script performs the following action:

- Grant the CREATE ANY DATABASE permission to [contoso\svc-ReportSvc$].

**NOTE**

These scripts will not configure SSRS. SSRS will get configured during deployment by the Service Fabric service (ReportingService) deployed to that node.

Instead, these scripts will grant the necessary permissions for the Service Fabric service (ReportingService) to carry out the necessary configuration.

---

**Step 11. Set up VMs**

1. Run the following command to export the scripts that must be run on each VM.

```
# Exports the script files to be executed on each VM into a directory VMs\<VMName>.\Export-Scripts.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

2. Download the following Microsoft Windows Installers (MSIs) into a file share that is accessible by all VMs.

<table>
<thead>
<tr>
<th>COMPONENT</th>
<th>DOWNLOAD LINK</th>
<th>EXPECTED FILE NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNAC – ODBC driver 13</td>
<td>ODBC Driver 13.1</td>
<td>msodbcsql.msi</td>
</tr>
<tr>
<td>SNAC – ODBC driver 17.5.x</td>
<td>ODBC Driver 17.5.2</td>
<td>msodbcsql_17.msi</td>
</tr>
<tr>
<td>SQL Server Management Studio 17.9.1</td>
<td>SSMS 17.9.1</td>
<td>SSMS-Setup-*__*.exe</td>
</tr>
<tr>
<td>Visual C++ Redistributable Packages for Microsoft Visual Studio 2013</td>
<td><a href="https://support.microsoft.com/help/3179560">https://support.microsoft.com/help/3179560</a></td>
<td>vc_redist_x64.exe</td>
</tr>
<tr>
<td>Visual C++ Redistributable Packages for Microsoft Visual Studio 2017</td>
<td>Go to <a href="https://lcs.dynamics.com/V2/SharedAssetLibrary">https://lcs.dynamics.com/V2/SharedAssetLibrary</a>, select Model as the asset type, and then select VC++ 17 Redistributables.</td>
<td>vc_redist.x64_14_16_27024.exe</td>
</tr>
<tr>
<td>COMPONENT</td>
<td>DOWNLOAD LINK</td>
<td>EXPECTED FILE NAME</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>---------------------------------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>The .NET Framework version 4.8 (CLR 4.0)</td>
<td><a href="https://dotnet.microsoft.com/download/thank-you/net48-offline">https://dotnet.microsoft.com/download/thank-you/net48-offline</a></td>
<td>ndp48-x86-x64-allos-enu.exe</td>
</tr>
<tr>
<td>The .NET Framework version 4.7.2 (CLR 4.0)</td>
<td><a href="https://dotnet.microsoft.com/download/thank-you/net472-offline">https://dotnet.microsoft.com/download/thank-you/net472-offline</a></td>
<td>ndp472-x86-x64-allos-enu.exe</td>
</tr>
</tbody>
</table>

**IMPORTANT**

- Make sure that the Management Studio setup is in the same language as the operating system of the target machine.
- Make sure that the installer files have the names that are specified in the “Expected file name” column of the preceding table. Rename any files that don’t have the expected name. Otherwise, you will encounter errors when you run the Configure-PreReqs.ps1 script.
- When you download VC++ 17 Redistributables, the executable file is inside the zip file.

Next, follow these steps for each VM, or use remoting from a single machine.

**NOTE**

- The following procedure requires execution on multiple VMs. However, to simplify the process, you can use the remoting scripts that are provided. These scripts let you run the required scripts from a single machine, such as the same machine that is used to run the \Export-Scripts.ps1 command. When the remoting scripts are available, they are declared after a # If Remoting comment in the Windows PowerShell sections. If you use the remoting scripts, you might not have to run the remaining scripts in a section. In these cases, see the section text.
- Remoting uses WinRM. In some cases, it requires that CredSSP be enabled. The remoting module enables and disables CredSSP on an execution-by-execution basis. We recommend that you disable CredSSP enabled when it isn’t used. Otherwise, there is a risk of credential theft. When you’ve completed the setup, see the Step 20. Tear down CredSSP if remoting was used section later in this topic.

1. Copy the contents of each infrastructure\VMs\<VMName> folder to the corresponding VM. (If the remoting scripts are used, they will automatically copy the contents to the target VMs.) Then run the following command as an administrator.

   ```powershell
   # Install prereq software on the VMs.
   # If remoting, execute
   # .\Configure-PreReqs-AllVMs.ps1 -MSIFilePath <share folder path of the MSIs> -ConfigurationFilePath .\ConfigTemplate.xml
   .\Configure-PreReqs.ps1 -MSIFilePath <path of the MSIs>
   ```
Step 12. Set up a standalone Service Fabric cluster

1. Download the Service Fabric standalone installation package to one of your Service Fabric nodes.

2. After the zip file is downloaded, select and hold (or right-click) it, and then select Properties. In the Properties dialog box, select the Unblock checkbox.

3. Copy the zip file to one of the nodes in the Service Fabric cluster, and unzip it. Make sure that the infrastructure folder has access to this folder.

4. Go to the infrastructure folder, and run the following command to generate the Service Fabric ClusterConfig.json file.

   ```
   .\New-SFClusterConfig.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -TemplateConfig <ServiceFabricStandaloneInstallerPath>\ClusterConfig.X509.MultiMachine.json
   ```

5. You might have to make additional modifications to your cluster configuration, based on your environment. For more information, see Step 1B: Create a multi-machine cluster, Secure a standalone cluster on Windows using X.509 certificates, and Create a standalone cluster running on Windows Server.

6. Copy the ClusterConfig.json file that is generated to `<ServiceFabricStandaloneInstallerPath>`.

7. Open Windows PowerShell in elevated mode, go to `<ServiceFabricStandaloneInstallerPath>`, and run the following command to test the ClusterConfig.json file.

   ```
   .\Test-D365FOConfiguration.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
   ```

   If you used remoting, be sure to run the cleanup steps after the setup is completed. For instructions, see the Step 20. Tear down CredSSP, if remoting was used section.
**Step 13. Configure LCS connectivity for the tenant**

An on-premises local agent is used to orchestrate deployment and servicing of Finance + Operations through LCS. To establish connectivity from LCS to the Finance + Operations tenant, you must configure a certificate that enables the local agent to act on behalf on your Azure AD tenant (for example, contoso.onmicrosoft.com).

Use the on-premises agent certificate that you acquired from a CA or the self-signed certificate that you generated by using scripts. The on-premises agent certificate can be reused across multiple sandbox and production environments per tenant.

Only user accounts that have the Global Administrator directory role can add certificates to authorize LCS. By default, the person who signs up for Microsoft 365 for your organization is the global administrator for the directory.

**IMPORTANT**
- You must configure the certificate exactly one time per tenant. All on-premises environments under the same tenant must use the same certificate to connect with LCS.
- If you run the script below on a server machine (for example, a machine that is running Windows Server 2019), you must temporarily turn off the Internet Explorer Enhanced Security Configuration. Otherwise, the content on the Azure sign-in page will be blocked.

1. Sign in to the customer’s Azure portal to verify that you have the Global Administrator directory role.
2. From the **infrastructure** folder, run the following commands to determine whether the certificate is already registered.

```powershell
# If you have issues downloading the Azure PowerShell Az module, run the following:
Install-Module Az
Import-Module Az
.Add-CertToServicePrincipal.ps1 -CertificateThumbprint 'OnPremLocalAgent Certificate Thumbprint' - Test
```

**IMPORTANT**

If you previously installed AzureRM, you should remove it, because it might not be compatible with any existing AzureRM installations in Windows PowerShell 5.1. For more information, see [Migrate Azure PowerShell from AzureRM to Az](#).

3. If the script indicates that the certificate isn't registered, run the following command.

```powershell
.Add-CertToServicePrincipal.ps1 -CertificateThumbprint 'OnPremLocalAgent Certificate Thumbprint'
```

**NOTE**

If you have multiple tenants that are associated with the login account, you can run the following command to pass the tenant ID as a parameter. In this way, you can ensure that the context is set to the correct tenant.

```powershell
```

---

**Step 14. Set up file storage**

You must set up the following SMB 3.0 file shares:

- A file share that stores user documents that are uploaded to AOS (for example, `\DAX7SQLAOFILE1\aos-storage`).
- A file share that stores the latest build and configuration files to orchestrate the deployment (for example, `\DAX7SQLAOFILE1\agent`).

**WARNING**

Keep this file share path as short as possible, to avoid exceeding the maximum path length on the files that will be put in the share.

For information about how to enable SMB 3.0, see [SMB Security Enhancements](#).
IMPORTANT

- Secure dialect negotiation can't detect or prevent downgrades from SMB 2.0 or 3.0 to SMB 1.0. Therefore, we strongly recommend that you disable the SMB 1.0 server. In this way, you can take advantage of the full capabilities of SMB encryption.
- To help ensure that your data is protected while it's at rest in your environment, you must enable BitLocker Drive Encryption on every machine. For information about how to enable BitLocker, see BitLocker: How to deploy on Windows Server 2012 and later.

1. On the file share machine, run the following command.

   ```
   Install-WindowsFeature -Name FS-FileServer -IncludeAllSubFeature -IncludeManagementTools
   ```

2. Set up the `\DAX7SQLAOFILE1\aos-storage` file share:
   a. In Server Manager, select File and Storage Services > Shares.
   b. Select Tasks > New Share to create a share. Name the new share `aos-storage`.
   c. Leave Allow caching of share selected.
   d. Select the Encrypt data access checkbox.
   e. Grant Modify permissions for every machine in the Service Fabric cluster except OrchestratorType.
   f. Grant Modify permissions for the user AOS domain user (`contoso\AXServiceUser`) and the gMSA user (`contoso\svc-AXSF$`).

   **NOTE**

   To add machines, you might have to enable Computers under Object Types. To add service accounts, you might have to enable Service Accounts under Object Types.

   If you are deploying with a recent base deployment where a gMSA account is used instead of the domain user, you can skip adding the AOSDomainUser account to the fileshare ACLs.

3. Set up the `\DAX7SQLAOFILE1\agent` file share:
   a. In Server Manager, select File and Storage Services > Shares.
   b. Select Tasks > New Share to create a share. Name the new share `agent`.
   c. Grant Full-Control permissions to the gMSA user for the local deployment agent (`contoso\svc-LocalAgent$`).
Step 15. Set up SQL Server

1. Install SQL Server with high availability, unless you're deploying in a sandbox environment, where one instance of SQL Server is sufficient. (Nevertheless, you might want to install SQL Server with high availability in sandbox environments to test high-availability scenarios.)

   **IMPORTANT**
   
   You must enable the SQL Server and Windows Authentication mode.

   You can install SQL Server with high availability either as SQL clusters that include a Storage Area Network (SAN) or in an Always-On configuration. Verify that the Database Engine, SSRS, Full-Text Search, and SQL Server Management Tools are already installed.

   **NOTE**
   
   Make sure that Always-On is set up as described in Select Initial Data Synchronization Page (Always On Availability Group Wizards), and follow the instructions in To Prepare Secondary Databases Manually.

2. Run the SQL service as either a domain user or a gMSA.

3. Get an SSL certificate from a CA to configure SQL Server for Finance + Operations. For testing purposes, you can create and use a certificate that is generated through AD CS. You will have to replace the computer name and domain name in the following examples.

   **AD CS certificate for an Always-On SQL availability group**

   If you're setting up testing certificates for Always-On, use the following remoting script. This script works like the manual script that follows.
#If you need to create self-signed certs
#. \New-SelfSigned-SQLCert-AllVMs.ps1 -SqlMachineNames SQL1,SQL2 -SqlListenerName SQL-LS -ProtectTo CONTOSO\dynuser -ConfigurationFilePath .\ConfigTemplate.xml
.. \New-ADCS-SQLCert-AllVMs.ps1 -SqlMachineNames SQL1,SQL2 -SqlListenerName SQL-LS -ProtectTo CONTOSO\dynuser

AD CS certificate for a single SQL availability group

#If you need to create self-signed certs
#. \New-SelfSigned-SQLCert-AllVMs.ps1 -SqlMachineNames SQL1 -ProtectTo CONTOSO\dynuser -ConfigurationFilePath .\ConfigTemplate.xml
.. \New-ADCS-SQLCert-AllVMs.ps1 -SqlMachineNames SQL1 -ProtectTo CONTOSO\dynuser

Manual AD CS steps for an Always-On SQL availability group or Windows Server Failover Clustering with SQL Server

For each node of the SQL cluster, follow these steps.

a. Run the following Windows PowerShell command on each of the SQL Server Always-On replicas.

```
#If you need to create self-signed certs
#. \New-SelfSigned-SQLCert-AllVMs.ps1 -SqlMachineNames SQL1,SQL2 -SqlListenerName SQL-LS -ProtectTo CONTOSO\dynuser -GenerateCertOnly
.. \New-ADCS-SQLCert-AllVMs.ps1 -SqlMachineNames SQL1,SQL2 -SqlListenerName SQL-LS -ProtectTo CONTOSO\dynuser -GenerateCertOnly
```

b. Grant certificate permissions to the account that is used to run the SQL service:

   a. Open the Certificate Manager tool (certlm.msc).
   b. Select and hold (or right-click) the certificate that was created, and then select Tasks > Manage Private Keys.
   c. Add the SQL Server service account, and grant it Read access.
   d. Enable ForceEncryption and the new certificate in SQL Server Configuration Manager:

      a. Open SQL Server Configuration Manager, expand SQL Server Network Configuration, select and hold (or right-click) Protocols for [server instance], and then select Properties.
      b. In the Properties dialog box, on the Certificate tab, in the Certificate field, select the desired certificate.
      c. On the Flags tab, in the ForceEncryption box, select Yes.
      d. Select OK to save your changes.
   d. Export the certificate (.cer file) from each SQL cluster node, and install it in the trusted root of each Service Fabric node. You will have at least two certificates for the Always-On cluster. However, you might have more if you have additional replicas.

e. Restart the SQL service.

**NOTE**

For more information, see How to enable SSL encryption for an instance of SQL Server by using Microsoft Management Console.
**IMPORTANT**

If you used remoting, be sure to run the cleanup steps after the setup is completed. For instructions, see the Step 20. Tear down CredSSP, if remoting was used section.

---

**Step 16. Configure the databases**

1. Sign in to LCS.

2. On the dashboard, select the **Shared asset library** tile.

3. Select **Model** as the asset type. Then, in the grid, select the data type for the release that you want, and download the zip file.

<table>
<thead>
<tr>
<th>RELEASE</th>
<th>DATABASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>On-premises Platform update 41</td>
<td>Dynamics 365 for Operations on-premises, Version 10.0.17 Demo Data</td>
</tr>
<tr>
<td>On-premises Platform update 41</td>
<td>Dynamics 365 for Operations on-premises, Version 10.0.17 Empty Data</td>
</tr>
</tbody>
</table>

4. The zip file contains a single backup (.bak) file. Select the file to download, based on your requirements.

5. After the zip file is downloaded, verify that it's unblocked. Select and hold (or right-click) the file, and then select **Properties**. In the **Properties** dialog box, select the **Unblock** checkbox.

6. Make sure that the database section in the `{infrastructure}\ConfigTemplate.xml` file is correctly configured with the following information:

   - The database name.
   - The database file and log settings. The database settings should not be lower than the default values that are specified.
   - The path of the backup file that you downloaded earlier. The default name of the Finance + Operations database is **AXDB**.

**IMPORTANT**

- The user who is running the SQL service and the user who is running the scripts should have **Read** access on the folder or share where the backup file is located.
- If an existing database already has the same name, it won't be overwritten.

7. Copy the `{infrastructure}` folder to the SQL Server machine. Then open Windows PowerShell in elevated mode, and go to the folder.

**Configure the OrchestratorData database**

- Run the following command.

  ```powershell
  .\Initialize-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName Orchestrator
  ```

  The `Initialize-Database.ps1` script performs the following actions:

  1. Create an empty database that is named **OrchestratorData**. This database is used by the on-premises local agent to orchestrate deployments.

  2. Grant **db_owner** permissions on the database to the local agent gMSA (svc-LocalAgent$).
Configure the Finance + Operations database

1. Run the following commands.

```
.Initialize-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName AOS
.Configure-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName AOS
.Reset-DatabaseUsers.ps1 -DatabaseServer '<FQDN of the SQL server>' -DatabaseName '<AX database name>'
```

The Initialize-Database.ps1 script performs the following actions:

a. Restore the database from the specified backup file.

b. Create a new user that SQL authentication is enabled for (axdbadmin).

c. Map users to database roles, based on the following table for the AXDB database.

<table>
<thead>
<tr>
<th>USER</th>
<th>TYPE</th>
<th>DATABASE ROLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>svc-AXSF$</td>
<td>gMSA</td>
<td>db_owner</td>
</tr>
<tr>
<td>svc-LocalAgent$</td>
<td>gMSA</td>
<td>db_owner</td>
</tr>
<tr>
<td>svc-FRPS$</td>
<td>gMSA</td>
<td>db_owner</td>
</tr>
<tr>
<td>svc-FRAS$</td>
<td>gMSA</td>
<td>db_owner</td>
</tr>
<tr>
<td>axdbadmin</td>
<td>SqlUser</td>
<td>db_owner</td>
</tr>
</tbody>
</table>

d. Map users to database roles, based on the following table for the TempDB database.

<table>
<thead>
<tr>
<th>USER</th>
<th>TYPE</th>
<th>DATABASE ROLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>svc-AXSF$</td>
<td>gMSA</td>
<td>db_datareader, db_datawriter, db_ddladmin</td>
</tr>
<tr>
<td>axdbadmin</td>
<td>SqlUser</td>
<td>db_datareader, db_datawriter, db_ddladmin</td>
</tr>
</tbody>
</table>

The Configure-Database.ps1 script performs the following actions:

a. Set READ COMMITTED_SNAPSHOT to ON.

b. Set ALLOW_SNAPSHOT_ISOLATION to ON.

c. Set the specified database file and log settings.

d. Grant the VIEW SERVER STATE permission to axdbadmin.

e. Grant the ALTER ANY EVENT SESSION permission to axdbadmin.

f. Grant the VIEW SERVER STATE permission to [contoso\svc-AXSF$].

g. Grant the ALTER ANY EVENT SESSION permission to [contoso\svc-AXSF$].

2. Run the following command to reset the database users.

```
.Reset-DatabaseUsers.ps1 -DatabaseServer '<FQDN of the SQL server>' -DatabaseName '<AX database name>'
```

Configure the Financial Reporting database

- Run the following command.
The Initialize-Database.ps1 script performs the following actions:

1. Create an empty database that is named **FinancialReporting**.
2. Map the users to database roles, based on the following table.

<table>
<thead>
<tr>
<th>USER</th>
<th>TYPE</th>
<th>DATABASE ROLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>svc-LocalAgent$</td>
<td>gMSA</td>
<td>db_owner</td>
</tr>
<tr>
<td>svc-FRPS$</td>
<td>gMSA</td>
<td>db_owner</td>
</tr>
<tr>
<td>svc-FRAS$</td>
<td>gMSA</td>
<td>db_owner</td>
</tr>
</tbody>
</table>

**Step 17. Encrypt credentials**

1. On any client machine, install the `axdataencipherment` certificate in the `LocalMachine\My` certificate store.
2. Grant the current user **Read** access to the private key of this certificate.
3. Create the **Credentials.json** file, as shown here.

```json
{
    "AosPrincipal": {
        "AccountPassword": "<encryptedDomainUserPassword>",
    },
    "AosSqlAuth": {
        "SqlUser": "<encryptedSqlUser>",
        "SqlPwd": "<encryptedSqlPassword>"
    }
}
```

- **AccountPassword** – The encrypted domain user password for the AOS domain user (`contoso\axserviceuser`).

**NOTE**

If you are deploying with a recent base deployment where a gMSA account is used instead of the domain user, leave the **AccountPassword** field blank. However, you need to ensure that it is present, as the installers will still look for it. We will address this in a future update.

- **SqlUser** – The encrypted SQL user (`axdbadmin`) that has access to the Finance + Operations database (`AXDB`).
- **SqlPassword** – The encrypted SQL password.
4. Copy the .json file to the SMB file share: `\AX7SQLAOFILE1\agent\Credentials\Credentials.json`.
5. Update the **Credentials.json** file with encrypted values.

# Service fabric API to encrypt text and copy it to the clipboard.
Invoke-ServiceFabricEncryptText -Text '<textToEncrypt>' -CertThumbprint '<AxDataEncipherment Thumbprint>' -CertStore -StoreLocation LocalMachine -StoreName My | Set-Clipboard
Step 18. Configure AD FS

Before you can complete this procedure, AD FS must be deployed on Windows Server. For information about how to deploy AD FS, see Deployment Guide Windows Server 2016 and 2012 R2 AD FS Deployment Guide.

Finance + Operations requires additional configuration of AD FS, beyond the default out-of-box configuration. The following Windows PowerShell commands must be run on the machine where the AD FS role service is installed. The user account must have enough permissions to administer AD FS. For example, the user must have a domain administrator account. For complex AD FS scenarios, consult your domain administrator.

1. Configure the AD FS identifier so that it matches the AD FS token issuer.

   This command is related to adding new users by using the **Import users** option on the **Users** page (**System administration** > **Users** > **Users**) in the Finance + Operations client.

   ```powershell
   $adfsProperties = Get-AdfsProperties
   Set-AdfsProperties -Identifier $adfsProperties.IdTokenIssuer
   ```

   **WARNING**

   If your AD FS is set up to work with Microsoft 365 (formerly Office 365) for single sign-on, this step will break that scenario.

   To ensure that the scenario continues to work, you can specify a deployment option to adapt your Dynamics 365 for Finance + Operations installation to that requirement. For more information, see AD FS Microsoft 365 compatibility.

2. You should disable Windows Integrated Authentication (WIA) for intranet authentication connections, unless you’ve configured AD FS for mixed environments. For more information about how to configure WIA so that it can be used with AD FS, see Configure browsers to use Windows Integrated Authentication (WIA) with AD FS.

   This command is related to using forms authentication upon sign-in to the Finance + Operations client. Other options, such as single sign-on, are not supported.

   ```powershell
   Set-AdFsGlobalAuthenticationPolicy -PrimaryIntranetAuthenticationProvider FormsAuthentication, MicrosoftPassportAuthentication
   ```

3. For sign-in, the user’s email address must be acceptable authentication input.

   This command is related to setting up email claims. Other options, such as transformation rules, might be available but require additional setup.

   ```powershell
   ```
Before AD FS can trust Finance + Operations for the exchange of authentication, various application entries must be registered under an AD FS application group in AD FS. To speed up the setup process and help reduce errors, you can use the Publish-ADFSApplicationGroup.ps1 script for registration. Copy this script and the D365FO-OP directory to a machine where the AD FS role service is installed. Then run the script by using a user account that has enough permissions to administer AD FS. (For example, use an administrator account.)

For more information about how to use the script, see the documentation that is listed in the script. Make a note of the client IDs that are specified in the output, because you will need this information in LCS later. If you lose the client IDs, sign in to the machine where AD FS is installed, open Server Manager, and go to Tools > AD FS Management > Application Groups > Microsoft Dynamics 365 for Operations On-premises. You can find the client IDs under the native applications.

NOTE

If you want to reuse your previously configured AD FS server for additional environments, see Reuse the same AD FS instance for multiple environments.

Finally, verify that you can access the AD FS OpenID configuration URL on a Service Fabric node of the AOSNodeType type. To do this check, try to open

```wells-configuration
https://<adfs-dns-name>/adfs/.well-known/openid-configuration
```

in a web browser. If you receive a message that states that the site isn't secure, you haven't added your AD FS SSL certificate to the Trusted Root Certification Authorities store. This step is described in the AD FS deployment guide. If you're using remoting, you can run the following command to install the certificate on all nodes in the Service Fabric cluster.
# If remoting, execute
\Install-ADFSCert-AllVMs.ps1 -ConfigurationFilePath .\ConfigTemplate.xml

If you can access the URL, a JavaScript Object Notation (JSON) file is returned. This file contains your AD FS configuration, and it will indicate that your AD FS URL is trusted.

You’ve now completed the setup of the infrastructure. The following sections describe how set up your connector and deploy your Finance + Operations environment in LCS.

**Step 19. Configure a connector and install an on-premises local agent**

1. Sign in to LCS, and open your on-premises implementation project.

2. Select the Menu button (sometimes referred to as the hamburger or the hamburger button), and then select **Project settings**.

3. Select **On-premises connectors**.

4. Select **Add** to create a new on-premises connector.

5. On the **1: Setup host infrastructure** tab, select **Download agent installer**.

6. After the zip file is downloaded, verify that it’s unblocked. Select and hold (or right-click) the file, and then select **Properties**. In the **Properties** dialog box, select the **Unblock** checkbox.

7. Unzip the agent installer on one of the Service Fabric nodes of the **OrchestratorType** type.

8. After the file is unzipped, go back to your on-premises connector in LCS.

9. On the **2: Configure agent** tab, select **Enter configuration**, and enter the configuration settings. To get the required values, run the following command on any machine that has the **infrastructure** folder and up-to-date configuration files.

   \Get-AgentConfiguration.ps1 -ConfigurationFilePath .\ConfigTemplate.xml

10. Save the configuration, and then select **Download configurations** to download the **localagent-config.json** configuration file.

11. Copy the **localagent-config.json** file to the machine where the agent installer package is located.

12. In a **Command Prompt** window, go to the folder that contains the agent installer, and run the following command.

   LocalAgentCLI.exe Install <path of config.json>

   **NOTE**
   The user who runs this command must have **db_owner** permissions on the OrchestratorData database.

13. After the local agent is successfully installed, go back to your on-premises connector in LCS.

14. On the **3: Validate setup** tab, select **Message agent** to test for LCS connectivity to your local agent. When a connection is successfully established, you will receive the following message: “Validation complete. Agent connection established.”

**Step 20. Tear down CredSSP, if remoting was used**

If you used any of the remoting scripts during setup, be sure to run the following command during breaks in the
setup process, or after the setup is completed.

\.
\Disable-CredSSP-AllVMs.ps1 -ConfigurationFilePath .\ConfigTemplate.xml

If the previous remoting Windows PowerShell window was accidentally closed, and CredSSP was left enabled, this command disables it on all the machines that are specified in the configuration file.

**Step 21. Deploy your Finance + Operations environment from LCS**

1. In LCS, open your on-premises implementation project.

2. Go to Environment > Sandbox, and select Configure. To get the required values, run the following command on the primary domain controller VM. That VM must have access to ADFS and the DNS server settings.

\.
\Get-DeploymentSettings.ps1 -ConfigurationFilePath .\ConfigTemplate.xml

3. For new deployments, select your environment topology, and then complete the wizard to start your deployment.

   During the preparation phase, LCS assembles the Service Fabric application packages for your environment. It then sends a message to the local agent to start deployment. You should notice that the environment state is **Preparing**.

   ![Sandbox Environment](image)

   **Sandbox**

   - Environment EN04 state: Preparing

   - Full details

4. Select Full details to open the environment details page. Notice that the upper-right corner of the page shows the environment status as **Preparing**.

   ![Environment Details](image)

   **EN04**

   - Environment updates

   - EN04 (Activity ID: 411d8d59-b62a-4d91-918d-db5950ba07fd)

   The local agent picks up the deployment request, starts the deployment, and communicates back to LCS when the environment is ready. When deployment is started, you should notice that the environment state is changed to **Deploying**.
5. Select Full details to open the environment details page. Notice that the upper-right corner of the page shows the environment status as Deploying.

6. If the deployment fails, the environment state is changed to Failed, and the Reconfigure button becomes available for the environment. Fix the underlying issue, select Reconfigure, update any configuration changes, and then select Deploy to retry the deployment.

For information about how to reconfigure an environment, see Reconfigure environments to take a new platform or topology.

The following illustration shows a successful deployment. Notice that the upper-right corner of the page shows the environment status as Deployed.
Step 22. Connect to your Finance + Operations environment

- In a web browser, go to `https://[yourD365FOdomain]/namespaces/AXSF`, where `yourD365FOdomain` is the domain name that you defined in the Step 1. Plan your domain name and DNS zones section earlier in this topic.

Known issues

When you run the New-D365FOGMSAAccounts cmdlet, you receive the following error message: "Key does not exist"

If you're creating and generating gMSA passwords in your domain for the first time, you must first create the Key Distribution Services KDS Root Key. For more information, see Create the Key Distribution Services KDS Root Key.

When you run the remoting script Configure-Prereqs-AllVms cmdlet, you receive the following error message: "The WinRM client cannot process the request"

Follow the instructions in the error message to enable the Allow delegation fresh credentials computer policy on all machines of the Service Fabric cluster.

When you Configure-Prereqs on servers of the MRType and ReportServerType types, you receive the following error message: "Install-WindowsFeature: The request to add or remove features on the specified server failed"

The .NET Framework version 3.5 is required on servers of the MRType and ReportServerType types. However, by default, source files for the .NET Framework version 3.5 aren't included in Windows Server 2016 installations. To work around the error, install the .NET Framework version 3.5. When you use Server Manager to manually add new features, specify the source files by using the `source` option.

When you run the Publish-ADFSApplicationGroup cmdlet, you receive the following error message: "MSIS7628: Scope names should be a valid Scope description name in AD FS configuration"

This error occurs because an OpenID allatclaims scope that D365FO-OP-ADFSApplicationGroup requires might be missing in some Windows Server 2016 installations. To work around the error, open Server Manager, go to Tools /> AD FS Management /> Service /> Scope Descriptions, and add the allatclaims scope description.

When you run the Publish-ADFSApplicationGroup cmdlet, you receive the following error message: "ADMIN0077: Access control policy does not exist: Permit everyone"

If AD FS is installed with a non-English version of Windows Server 2016, the Permit everyone access control
policy is created in the local language. Invoke the cmdlet in the following way to specify the `AccessControlPolicyName` parameter.

```
.

Additional resources

- Apply updates to on-premises deployments
- Redeploy on-premises environments
- Configure document management
- Import Electronic reporting (ER) configurations
- Document generation, publishing, and printing in on-premises deployments
- Configure proxies for on-premises environments
- Set up technical support for Finance and Operations apps
- Client internet connectivity
This topic provides information about how to plan, set up, and deploy Dynamics 365 Finance + Operations (on-premises) with Platform update 12-40.

The Local Business Data Yammer group is available. You can post questions or feedback you may have about the on-premises deployment there.

If you have questions or feedback about the content in this topic, please post them in the Comments section at the bottom of this page.

Finance + Operations components

The Finance + Operations application consists of three main components:

- Application Object Server (AOS)
- Business Intelligence (BI)
- Financial Reporting/Management Reporter

These components depend on the following system software:

- Microsoft Windows Server 2016 (only English OS installations are supported)
- Microsoft SQL Server 2016 SP1 and SP2 (from Platform update 33), which has the following features:
  - Full-text index search is enabled.
  - SQL Server Reporting Services (SSRS) - This is deployed on BI virtual machines.
  - SQL Server Integration Services (SSIS) - This is deployed on AOS virtual machines.

- SQL Server Management Studio
- Standalone Microsoft Azure Service Fabric
- Microsoft Windows PowerShell 5.0 or later
- Active Directory Federation Services (AD FS) on Windows Server 2016
- Domain controller
Lifecycle Services

Finance + Operations bits are distributed through Microsoft Dynamics Lifecycle Services (LCS). Before you can deploy, you must purchase license keys through the Enterprise Agreements channel and set up an on-premises project in LCS. Deployments can be initiated only through LCS. For more information about how to set up on-premises projects in LCS, see Set up on-premises projects in Lifecycle Services (LCS).

Authentication

The on-premises application works with AD FS. To interact with LCS, you must also configure Azure Active Directory (AAD). To complete the deployment and configure the LCS Local agent, you will need AAD. If you do not already have an AAD tenant, you can get one for free by using one of the options provided by AAD. For more information, see How to get an Azure Active Directory tenant.

Standalone Service Fabric

Finance + Operations uses standalone Service Fabric. For more information, see the Service Fabric documentation.

Setup of Finance + Operations will deploy a set of applications inside Service Fabric (SF). During deployment, each node in the cluster will be defined via configuration to have one of the following node types:

- **AOSNodeType** - Hosts the application object server (business logic).
- **OrchestratorType** - Functions as Service Fabric primary nodes, and hosts deployment- and servicing logic.
- **ReportServerType** - Hosts SSRS and reporting logic.
- **MRType** - Hosts management reporting logic.

Infrastructure

Finance + Operations falls under the standard Microsoft support policy about operation on non-Microsoft virtualization platforms, specifically VMware. For more information, see Support policy for Microsoft software. In short, we support our products in this environment. However, if we are asked to investigate an issue, we might first ask the customer to reproduce the issue without the virtualization platform or on the Microsoft virtualization platform.

If you are using VMWare, you must implement the fixes that are documented on the following web pages:

- **After upgrading a virtual machine to hardware version 11, network dependent workloads experience performance degradation (2129176)**
- **Several issues with vmxnet3 virtual adapter**
The hardware configuration includes the following components:

- Standalone Service Fabric cluster that is based on Windows Server 2016 virtual machines (VMs)
- Microsoft SQL Server (both Clustered SQL and Always-On are supported)
- AD FS for authentication
- Server Message Block (SMB) version 3 file share for storage
- Optional: Microsoft Office Server 2017

For more information, see [System requirements for on-premises deployments](#).

**Hardware layout**

Plan your infrastructure and Service Fabric cluster based on the recommended sizing in [Hardware sizing requirements for on-premises environments](#). For more information about how to plan the Service Fabric cluster, see [Plan and prepare your Service Fabric standalone cluster deployment](#).

The following table shows an example of a hardware layout. This example is used throughout this topic to illustrate the setup. You will need to replace the machine names and IP addresses given in the following instructions with the names and IP addresses for the machines in your environment.

**NOTE**

The Primary node of the Service Fabric cluster must have at least three nodes. In this example, **OrchestratorType** is designated as the Primary node type.

<table>
<thead>
<tr>
<th>MACHINE PURPOSE</th>
<th>SF NODE TYPE</th>
<th>MACHINE NAME</th>
<th>IP ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain controller</td>
<td></td>
<td>DAX7SQLAOADC1</td>
<td>10.179.108.2</td>
</tr>
<tr>
<td>AD FS</td>
<td></td>
<td>DAX7SQLAOADFS1</td>
<td>10.179.108.3</td>
</tr>
<tr>
<td>File server</td>
<td></td>
<td>DAX7SQLAOFILE1</td>
<td>10.179.108.4</td>
</tr>
<tr>
<td>SQL Always-On cluster</td>
<td></td>
<td>DAX7SQLAOSQLA01</td>
<td>10.179.108.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DAX7SQLAOSQLA02</td>
<td>10.179.108.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DAX7SQLAOSQLA</td>
<td>10.179.108.9</td>
</tr>
<tr>
<td>Client</td>
<td></td>
<td>SQLAOCLIENT1</td>
<td>10.179.108.11</td>
</tr>
<tr>
<td>AOS 1</td>
<td>AOSNodeType</td>
<td>SQLAOSF1AOS1</td>
<td>10.179.108.12</td>
</tr>
<tr>
<td>AOS 2</td>
<td>AOSNodeType</td>
<td>SQLAOSF1AOS2</td>
<td>10.179.108.13</td>
</tr>
<tr>
<td>AOS 3</td>
<td>AOSNodeType</td>
<td>SQLAOSF1AOS3</td>
<td>10.179.108.14</td>
</tr>
<tr>
<td>Orchestrator 1</td>
<td>OrchestratorType</td>
<td>SQLAOSF1ORCH1</td>
<td>10.179.108.15</td>
</tr>
<tr>
<td>MACHINE PURPOSE</td>
<td>SF NODE TYPE</td>
<td>MACHINE NAME</td>
<td>IP ADDRESS</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------</td>
<td>-----------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Orchestrator 2</td>
<td>OrchestratorType</td>
<td>SQLAOSF1ORCH2</td>
<td>10.179.108.16</td>
</tr>
<tr>
<td>Orchestrator 3</td>
<td>OrchestratorType</td>
<td>SQLAOSF1ORCH3</td>
<td>10.179.108.17</td>
</tr>
<tr>
<td>Management Reporter node</td>
<td>MRType</td>
<td>SQLAOSMR1</td>
<td>10.179.108.18</td>
</tr>
<tr>
<td>SSRS node</td>
<td>ReportServerType</td>
<td>SQLAOSFBIN1</td>
<td>10.179.108.10</td>
</tr>
</tbody>
</table>

### Setup

#### Prerequisites

Before you start the setup, the following prerequisites must be in place. The setup of these prerequisites is out of scope for this document.

- Active Directory Domain Services (AD DS) must be installed and configured in your network.
- AD FS must be deployed.
- SQL Server 2016 SP2 must be installed on the SSRS machines.
- SQL Server Reporting Services 2016 must be installed in **Native** mode on the SSRS machines.

The following prerequisite software is installed on the VMs by the infrastructure setup scripts downloaded from LCS.

<table>
<thead>
<tr>
<th>NODE TYPE</th>
<th>COMPONENT</th>
<th>DETAILS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AOS</td>
<td>SNAC – ODBC driver 13</td>
<td>ODBC driver 13.1</td>
</tr>
<tr>
<td>AOS</td>
<td>SNAC – ODBC driver 17</td>
<td>This driver is needed for upgrading to PU15 or higher: <a href="https://aka.ms/downloadmsodbcsql">https://aka.ms/downloadmsodbcsql</a></td>
</tr>
<tr>
<td>AOS</td>
<td>The Microsoft .NET Framework version 4.0–4.6 (CLR 4.0)</td>
<td>Windows features: NET-Framework-45-Features, NET-Framework-45-Core, NET-Framework-45-ASPNET, NET-WCF-Services45, NET-WCF-TCP-PortSharing45</td>
</tr>
<tr>
<td>AOS</td>
<td>The Microsoft .NET Framework version 4.7.2 (CLR 4.0)</td>
<td><a href="https://dotnet.microsoft.com/downloadd/than-kyou/net472-offline">https://dotnet.microsoft.com/downloadd/than-kyou/net472-offline</a></td>
</tr>
<tr>
<td>NODE TYPE</td>
<td>COMPONENT</td>
<td>DETAILS</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>AOS</td>
<td>SQL Server Management Studio 17.2</td>
<td><a href="https://go.microsoft.com/fwlink/?linkid=854085">https://go.microsoft.com/fwlink/?linkid=854085</a></td>
</tr>
<tr>
<td>BI</td>
<td>.NET Framework version 4.0–4.6 (CLR 4.0)</td>
<td>Windows features: NET-Framework-45-Features, NET-Framework-45-Core, NET-Framework-45-ASPNET, NET-WCF-Services45, NET-WCF-TCP-PortSharing45</td>
</tr>
<tr>
<td>BI</td>
<td>The Microsoft .NET Framework version 4.7.2 (CLR 4.0)</td>
<td><a href="https://dotnet.microsoft.com/download/thank-you/net472-offline">https://dotnet.microsoft.com/download/thank-you/net472-offline</a></td>
</tr>
<tr>
<td>BI</td>
<td>SQL Server Management Studio 17.2</td>
<td><a href="https://go.microsoft.com/fwlink/?linkid=854085">https://go.microsoft.com/fwlink/?linkid=854085</a></td>
</tr>
<tr>
<td>MR</td>
<td>The Microsoft .NET Framework version 4.7.2 (CLR 4.0)</td>
<td><a href="https://dotnet.microsoft.com/download/thank-you/net472-offline">https://dotnet.microsoft.com/download/thank-you/net472-offline</a></td>
</tr>
<tr>
<td>ORCH</td>
<td>The Microsoft .NET Framework version 4.0–4.8 (CLR 4.0)</td>
<td><a href="https://dotnet.microsoft.com/download/thank-you/net48-offline">https://dotnet.microsoft.com/download/thank-you/net48-offline</a></td>
</tr>
</tbody>
</table>

**Overview**

The following steps must be completed to set up the infrastructure for Finance + Operations. Reading all the steps before you begin will make it easier to plan your setup.

1. Plan your domain name and DNS zones
2. Plan and acquire your certificates
1. Plan your domain name and DNS zones
2. Plan and acquire your certificates
3. Plan your users and service accounts
4. Create DNS zones, and add A records
5. Join VMs to the domain
6. Download setup scripts from LCS
7. Describe your configuration
8. Configure certificates
9. Setup VMs
10. Set up a standalone Service Fabric cluster
11. Configure LCS connectivity for the tenant
12. Set up file storage
13. Set up SQL Server
14. Configure the databases
15. Encrypt credentials
16. Set up SSIS
17. Set up SSRS
18. Configure AD FS
19. Configure a connector and install an on-premises local agent
20. Tear down CredSSP, if remoting was used
21. Deploy your Finance + Operations environment from LCS
22. Connect to your Finance + Operations environment

1. Plan your domain name and DNS zones

We recommend that you use a publicly registered domain name for your production installation of AOS. In that way, the installation can be accessed outside the network, if outside access is required.

For example, if your company's domain is contoso.com, your zone for Finance + Operations might be d365ffo.onprem.contoso.com, and the host names might be as follows:

- ax.d365ffo.onprem.contoso.com for AOS machines
- sf.d365ffo.onprem.contoso.com for the Service Fabric cluster

2. Plan and acquire your certificates

Secure Sockets Layer (SSL) certificates are required in order to secure a Service Fabric cluster and all the applications that are deployed. For your production and sandbox workloads, we recommend that you acquire certificates from a certificate authority (CA) such as DigiCert, Comodo, Symantec, GoDaddy, or GlobalSign. If your domain is set up with Active Directory Certificate Services (AD CS), you can create the certificates through AD CS. Each certificate must contain a private key that was created for key exchange, and it must be exportable to a Personal Information Exchange (.pfx) file.

Self-signed certificates can be used only for testing purposes. For convenience, the setup scripts provided in LCS include scripts that generate and export self-signed certificates. If you are using self-signed scripts, you will be instructed to run the creation scripts in later steps. As we've mentioned, these certificates can be used for testing purposes only.

Recommended settings for certificates are:

- Signature algorithm: sha256RSA
- Signature hash algorithm: sha256
- Public key: RSA (2048 bits)
- Thumbprint algorithm: sha1
<table>
<thead>
<tr>
<th>Purpose</th>
<th>Explanation</th>
<th>Additional Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Server SSL certificate</td>
<td>This certificate is used to encrypt data that is transmitted across a network between an instance of SQL Server and a client application.</td>
<td>The domain name of the certificate should match the fully qualified domain name (FQDN) of the SQL Server instance or listener. For example, if the SQL listener is hosted on the machine DAX7SQLAOSQLA, the certificate's DNS name is DAX7SQLAOSQLA.contoso.com. CN: DAX7SQLAOSQLA.contoso.com DNS Name: DAX7SQLAOSQLA.contoso.com</td>
</tr>
<tr>
<td>Service Fabric Server certificate</td>
<td>This certificate is used to help secure the node-to-node communication between the Service Fabric nodes. This certificate is also used as the Server certificate that is presented to the client that connects to the cluster.</td>
<td>For this certificate you can also use SSL wild card certificate of your domain. For example, *.contoso.com. This is explained in more details below the table. Otherwise, use the following values: CN: sfd365ffo.onprem.contoso.com DNS Name: sfd365ffo.onprem.contoso.com</td>
</tr>
<tr>
<td>Service Fabric Client certificate</td>
<td>This certificate is used by clients to view and manage the Service Fabric cluster.</td>
<td>CN: client.d365ffo.onprem.contoso.com DNS Name: client.d365ffo.onprem.contoso.com</td>
</tr>
<tr>
<td>Encipherment Certificate</td>
<td>This certificate is used to encrypt sensitive information such as the SQL Server password and user account passwords.</td>
<td>The certificate must be created by using the provider <strong>Microsoft Enhanced Cryptographic Provider v1.0</strong>. The certificate key usage must include Data Encipherment (10) and should not include Server authentication or Client authentication. For more information, see <a href="#">Managing secrets in Service Fabric applications</a>. CN: axdataenciphermentcert DNS Name: axdataenciphermentcert</td>
</tr>
<tr>
<td>PURPOSE</td>
<td>EXPLANATION</td>
<td>ADDITIONAL REQUIREMENTS</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| AOS SSL Certificate                | This certificate is used as the Server certificate that is presented to the client for the AOS website. It's also used to enable Windows Communication Foundation (WCF)/Simple Object Access Protocol (SOAP) certificates. | You can use the same wild card certificate that you used as the Service Fabric Server certificate. Otherwise, use the following values:  
CN: ax.d365ffo.onprem.contoso.com  
DNS Name: ax.d365ffo.onprem.contoso.com |
| Session Authentication certificate | This certificate is used by AOS to help secure a user's session information. | This certificate is also the File Share certificate that will be used at the time of deployment from LCS.  
CN: SessionAuthentication  
DNS Name: SessionAuthentication |
| Data Encryption certificate        | This certificate is used by the AOS to encrypt sensitive information.       | This must be created using the provider Microsoft Enhanced RSA and AES Cryptographic Provider.  
CN: DataEncryption  
DNS Name: DataEncryption |
| Data Signing certificate           | This certificate is used by the AOS to encrypt sensitive information.       | This is separate from the Data Encryption certificate and must be created using the provider Microsoft Enhanced RSA and AES Cryptographic Provider.  
CN: DataSigning  
DNS Name: DataSigning |
| Financial Reporting client certificate | This certificate is used to help secure the communication between the Financial Reporting services and the AOS. | CN: FinancialReporting  
DNS Name: FinancialReporting |
| Reporting certificate              | This certificate is used to help secure the communication between SSRS and the AOS. | Do not reuse the Financial Reporting Client certificate.  
CN: ReportingService  
DNS Name: ReportingService |
On-Premises local agent certificate

**PURPOSE**
This certificate is used to help secure the communication between a local agent that is hosted on-premises and on LCS. This certificate enables the local agent to act on behalf of your Azure AD tenant, and to communicate with LCS to orchestrate and monitor deployments.

**Note:** Only 1 on-premises local agent certificate is needed for a tenant.

**EXPLANATION**

CN: OnPremLocalAgent
DNS Name: OnPremLocalAgent

**ADDITIONAL REQUIREMENTS**

SSL wild card certificate of your domain can be used to combine Service Fabric Server certificate and AOS SSL certificate.

The following is an example of a Service Fabric Server certificate combined with an AOS SSL certificate.

**Subject name**

| CN = *.d365ffo.onprem.contoso.com |

**Subject alternative names**

| DNS Name=ax.d365ffo.onprem.contoso.com |
| DNS Name=sf.d365ffo.onprem.contoso.com |
| DNS Name=*.d365ffo.onprem.contoso.com |

**NOTE**
The wild card certificate allows you to secure only the first-level subdomain of the domain to which it is issued.

3. **Plan your users and service accounts**

You must create several user or service accounts for Finance + Operations to work. You must create a combination of group managed service accounts (gMSAs), domain accounts, and SQL accounts. The following table shows the user accounts, their purpose, and example names that will be used in this topic.

<table>
<thead>
<tr>
<th>USER ACCOUNT</th>
<th>TYPE</th>
<th>PURPOSE</th>
<th>USER NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Financial Reporting Application Service Account</td>
<td>gMSA</td>
<td></td>
<td>Contoso\svc-FRAS$</td>
</tr>
<tr>
<td>Financial Reporting Process Service Account</td>
<td>gMSA</td>
<td></td>
<td>Contoso\svc-FRPS$</td>
</tr>
<tr>
<td>Financial Reporting Click Once Designer Service Account</td>
<td>gMSA</td>
<td></td>
<td>Contoso\svc-FRCO$</td>
</tr>
</tbody>
</table>
This user should be created for future proofing. We plan to enable AOS to work with the gMSA in upcoming releases. By creating this user at the time of setup, you will help to ensure a seamless transition to the gMSA.*

<table>
<thead>
<tr>
<th>USER ACCOUNT</th>
<th>TYPE</th>
<th>PURPOSE</th>
<th>USER NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>AOS Service Account</td>
<td>gMSA</td>
<td>This user should be created for future proofing. We plan to enable AOS to work with the gMSA in upcoming releases. By creating this user at the time of setup, you will help to ensure a seamless transition to the gMSA.*</td>
<td>Contoso\svc-AXSF$</td>
</tr>
<tr>
<td>AOS Service Account</td>
<td>Domain account</td>
<td>AOS uses this user in the general availability (GA) release.</td>
<td>Contoso\AXServiceUser</td>
</tr>
<tr>
<td>AOS SQL DB Admin user</td>
<td>SQL user</td>
<td>Finance + Operations uses this user to authenticate with SQL**. This user will also be replaced by the gMSA user in upcoming releases***.</td>
<td>AXDBAdmin</td>
</tr>
<tr>
<td>Local Deployment Agent Service Account</td>
<td>gMSA</td>
<td>This account is used by the local agent to orchestrate the deployment on various nodes.</td>
<td>Contoso\Svc-LocalAgent$</td>
</tr>
</tbody>
</table>

* These accounts should not have their regional settings changed. They should have the default EN-US region settings.

** If the password of the SQL user contains special characters, you might encounter issues during deployment.

*** The SQL user name and password for SQL authentication are secured because they are encrypted and stored in the file share.

4. Create DNS zones and add A records

DNS is integrated with AD DS, and lets you organize, manage, and find resources in a network. The following instructions provide steps to create a DNS forward lookup zone and A records for the AOS host name and Service Fabric cluster. In this example setup, the DNS zone name is d365ffo.onprem.contoso.com, and the A records/host names are as follows:

- `ax.d365ffo.onprem.contoso.com` for AOS machines
- `sf.d365ffo.onprem.contoso.com` for the Service Fabric cluster

Add a DNS zone

Use the following procedure to add a DNS zone.

1. Sign in to the domain controller machine, select **Start**, and start DNS Manager by typing `dnsmgmt.msc` and selecting the `dnsmgmt (DNS)` application.
2. Right-click the domain controller name in the console tree, and then select **New Zone > Next**.
3. Select **Primary Zone**.
4. Leave the `Store the zone in Active Directory (available only if the DNS Server is a writeable domain controller)` check box selected, and then select **Next**.
5. Select **To all DNS Servers running on Domain Controllers in this domain: Contoso.com**, and then select **Next**.
6. Select **Forward Lookup Zone**, and then select **Next**.
7. Enter the zone name for your setup, and then select **Next**. For example, enter \d365ffo.onprem.contoso.com.  
8. Select **Do not allow dynamic updates**, and then select **Next**.  
9. Select **Finish**.

**Set up an A record for AOS**

In the new DNS zone, create one A record that is named **ax.d365ffo.onprem.contoso.com** for each Service Fabric cluster node of the **AOSNodeType** type. Don't create A records for the other node types.

1. Find the newly created zone under the **Forward Lookup Zones** folder in DNS Manager.  
2. Right-click the new zone, and then select **New Host**.  
3. Enter the name and IP address of the Service Fabric node. (For example, enter **ax** as the name and enter **10.179.108.12** as the IP address.) Select **Add Host**.  
4. Do not select either check box.  
5. Repeat steps 1-4 for each AOS node.

**Set up an A record for the orchestrator**

In the new DNS zone, create an A record that is named **sf.d365ffo.onprem.contoso.com** for each Service Fabric cluster node of the **OrchestratorType** type. Don't create A records for the other node types.

1. Right-click the new zone, and then select **New Host**.  
2. Enter the name and IP address of the Service Fabric node. (For example, enter **sf** as the name and enter **10.179.108.15** as the IP address.) Select **Add Host**.  
3. Do not select either check box.  
4. Repeat for each Orchestrator node.

**5. Join VMs to the domain**

Join each VM to the domain by completing the steps in the **Join a Computer to a Domain** document. Alternatively, use the following Windows PowerShell script.

```powershell
$domainName = Read-Host -Prompt 'Specify domain name (ex: contoso.com)'
Add-Computer -DomainName $domainName -Credential (Get-Credential -Message 'Enter domain credential')
```

**IMPORTANT**

You must restart the VMs after you join them to the domain.

**6. Download setup scripts from LCS**

We have provided several scripts to help improve the setup experience. Follow these steps to download the setup scripts from LCS.

**IMPORTANT**

The scripts must be executed from a computer in the same domain that the on-premises infrastructure is in.

1. Sign in to **LCS**.  
2. On the dashboard, select the **Shared asset library** tile.  
3. On the **Model** tab, in the grid, select the **Dynamics 365 for Operations on-premises - Deployment scripts** row.  
4. Select **Versions**, and then download the latest version of the zip file for the scripts.
5. Right-click the zip file, and then select **Properties**. In the dialog box, select the **Unblock** check box.
6. Copy the zip file to the machine that will be used to execute the scripts.
7. Unzip the files into a folder that is named **infrastructure**.

**NOTE**

If you need the older version for Platform update 8 or Platform update 11, download version 1.

---

**7. Describe your configuration**

The infrastructure setup scripts use the following configuration files to drive the setup.

- `infrastructure\ConfigTemplate.xml`
- `infrastructure\D365FO-OP\NodeTopologyDefinition.xml`
- `infrastructure\D365FO-OP\DatabaseTopologyDefinition.xml`

**NOTE**

Configuration files must be updated based on your environment for the setup scripts to work correctly. Be sure to update these files with the proper computer names, IP addresses, service accounts, and domain based on your setup.

---

`infrastructure\ConfigTemplate.xml` describes:

- Service Accounts that are needed for the application to operate
- Certificates necessary for securing communications
- Database configuration
- Service Fabric cluster configuration

**IMPORTANT**

Make sure that there are three fault domains for OrchestratorType when you configure Service Fabric cluster.
Make sure that no more than one type of node is deployed in a single machine when you configure Service Fabric cluster.

---

For each Service Fabric node type, `infrastructure\D365FO-OP\NodeTopologyDefinition.xml` describes:

- The mapping between each node type and the application, domain and service accounts, and certificates.
- Whether to enable the UAC.
- Prerequisites for Windows features and system software.
- Whether strong name validation should be enabled.
- List of firewall ports to be opened.

For each database, `infrastructure\D365FO-OP\DatabaseTopologyDefinition.xml` describes:

- The database settings.
- The mappings between users and roles.

Create **gMSA and domain user accounts**
1. Navigate to the machine that has the unzipped infrastructure scripts in the **infrastructure** folder.

2. Copy the **infrastructure** folder to the domain controller machine.

3. Start Windows PowerShell in elevated mode, change the directory to the **infrastructure** folder, and run the following commands.

   ```powershell
   Import-Module .\D365FO-OP\D365FO-OP.psd1
   New-D365FOGMSAAccounts -ConfigurationFilePath .\ConfigTemplate.xml
   Update-D365FOGMSAAccounts -ConfigurationFilePath .\ConfigTemplate.xml
   ```

4. Add the AOS Service Accounts, **Contoso\svc-AXSF$** and **Contoso\AXServiceUser** to the local administrators group for all AOS machines. For more information, see [Add a member to local group](#).

5. If you must make changes to accounts or machines, update the `ConfigTemplate.xml` file in the original **infrastructure** folder, copy it to this machine and then run the following script.

   ```powershell
   Update-D365FOGMSAAccounts -ConfigurationFilePath .\ConfigTemplate.xml
   ```

8. **Configure certificates**

1. Navigate to the machine that has the **infrastructure** folder.

2. Generate certificates:
   a. If you must generate self-signed certificates:
      a. Set the `generateSelfSignedCert` attribute to **true**. Only set this for the certificates that you need to generate.
      b. Run the following command. The script will create the certificates. Put the certificates in the `CurrentUser\My certificate store` on the machine, and update the thumbprints in the XML file.

         ```powershell
         # Create self-signed certs
         .\New-SelfSignedCertificates.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
         ```

   b. If you want to generate Active Directory Certificate Services (AD CS) certificates:
      a. Set the `generateADCSCert` attribute to **false** for the certificates that you don't want generated.
      b. Run the following commands. The script will create the certificate templates in AD CS. Generate the certificates from the templates, place the certificates in the `CurrentUser\My certificate store` on the machine, and update the thumbprints in the XML file.

         ```powershell
         .\New-ADSCertificates.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -CreateTemplates
         .\New-ADSCertificates.ps1 -(ConfigurationFilePath .\ConfigTemplate.xml
         ```

   **NOTE**

   The AD CS scripts need to run on a Domain Controller, or a Windows Server with Remote Server Admin Tools installed.
3. If you're using SSL certificates that were already generated, skip the certificate generation and update the thumbprints in the configTemplate.xml file. The certificates need to be installed in the CurrentUser\My store and their private keys must be exportable.

**WARNING**

Because of a leading not-printable special character, which is difficult to determine when present, the cert manager should not be used to copy thumbprints. If the not-printable special character is present, you will get the error, **X509 certificate not valid**. To retrieve the thumbprints, see results from PowerShell commands or run the following commands in PowerShell.

```
dir cert:\CurrentUser\My
dir cert:\LocalMachine\My
dir cert:\LocalMachine\Root
```

4. Specify a semi-colon separated list of users or groups in the **ProtectTo** tag for each certificate. Only Active directory users and groups specified in the **ProtectTo** tag will have permissions to import the certificates that are exported using the scripts. Passwords are not supported by the script to protect the exported certificates.

5. Export the certificates into .pfx files. As part of the export, this script will check that your certificates have the correct cryptographic provider set.

```
# Exports Pfx files into a directory VMs\<VMName>, all the certs will be written to infrastructure\Certs folder.
.
```

9. **Setup VMs**

1. Export the scripts that must be run on each VM.

```
# Exports the script files to be execute on each VM into a directory VMs\<VMName>.
.
```

2. Download the following Microsoft Windows Installers (MSIs) into a file share that is accessible by all VMs.

<table>
<thead>
<tr>
<th>COMPONENT</th>
<th>DOWNLOAD LINK</th>
<th>EXPECTED FILE NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNAC – ODBC driver 13</td>
<td>ODBC driver 13.1</td>
<td>msodbcsql.msi</td>
</tr>
<tr>
<td>SNAC – ODBC driver 17</td>
<td><a href="https://aka.ms/downloadmsodbcsql">https://aka.ms/downloadmsodbcsql</a></td>
<td>msodbcsql_17.msi</td>
</tr>
<tr>
<td>Microsoft SQL Server Management Studio 17.5</td>
<td>SSMS 17.5</td>
<td>SSMS-Setup-*.exe</td>
</tr>
<tr>
<td>Microsoft Visual C++ Redistributable Packages for Microsoft Visual Studio 2013</td>
<td><a href="https://support.microsoft.com/help/3179560">https://support.microsoft.com/help/3179560</a></td>
<td>vcredist_x64.exe</td>
</tr>
<tr>
<td>Microsoft Visual C++ Redistributable Packages for Microsoft Visual Studio 2017</td>
<td>Go to <a href="https://lcs.dynamics.com/V2/SharedAssetLibrary">https://lcs.dynamics.com/V2/SharedAssetLibrary</a>, select Model as the asset type, and then select VC++ 17 Redistributables.</td>
<td>vc_redist.x64_14_16_27024.exe</td>
</tr>
<tr>
<td>COMPONENT</td>
<td>DOWNLOAD LINK</td>
<td>EXPECTED FILE NAME</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>The Microsoft .NET Framework version 4.8 (CLR 4.0)</td>
<td><a href="https://dotnet.microsoft.com/download/thank-you/net48-offline">https://dotnet.microsoft.com/download/thank-you/net48-offline</a></td>
<td>ndp48-x86-x64-allos-enu.exe</td>
</tr>
<tr>
<td>The Microsoft .NET Framework version 4.7.2 (CLR 4.0)</td>
<td><a href="https://dotnet.microsoft.com/download/thank-you/net472-offline">https://dotnet.microsoft.com/download/thank-you/net472-offline</a></td>
<td>ndp472-x86-x64-allos-enu.exe</td>
</tr>
</tbody>
</table>

**IMPORTANT**
- Make sure the Microsoft SQL Server Management Studio setup is in the same language as the operating system of the target machine.
- Make sure that the installer files have the names that are specified in the "Expected file name" column of the preceding table.
- You may need to rename some of the downloads if the "Expected file name" is different. Failure to do so will result in errors when running the "Configure-PreReqs.ps1" script.
- When you download VC++ 17 Redistributables, the executable file is inside the zip file.

**NOTE**

Follow these steps for each VM, or use remoting from a single machine

The following section requires execution on multiple VMs. This process can be eased by using the supplied remoting scripts, which provide the option of running the necessary scripts from a single machine, such as the same machine used to execute .\Export-Scripts.ps1. The remoting scripts, when available, are declared after a "# If Remoting " comment in the PowerShell sections. When the remoting scripts are used, you may not need to execute the remaining scripts in a section, please see the section text for cases such as that. Remoting uses WinRM and requires CredSSP to be enabled in certain cases. The enabling and disabling of CredSSP is handled by the remoting module on a per-execution basis. Keeping CredSSP enabled when it is not in use is not advised, as it introduces security risks in the shape of credential theft. See the Tear down CredSSP section when you are finished setting up.

1. Copy the contents of each infrastructure\VMs<VMName> folder into the corresponding VM (if remoting scripts are used, they will automatically copy the content to the target VMs), and then run the following scripts as an Administrator.

```
# Install pre-req software on the VMs.

# If Remoting, execute
# .\Configure-PreReqs-AllVMs.ps1 -MSIFilePath <share folder path of the MSIs> -ConfigurationFilePath .\ConfigTemplate.xml

.\Configure-PreReqs.ps1 -MSIFilePath <path of the MSIs>
```
10. Set up a standalone Service Fabric cluster

1. Download the Service Fabric standalone installation package onto one of your Service Fabric nodes. After the zip file is downloaded, unblock it by right-clicking the zip file and then selecting Properties. In the dialog box, select the Unblock check box in the lower right.

2. Copy the zip file to one of the nodes in the Service Fabric cluster, and unzip it. Ensure the infrastructure folder has access to this folder.

3. Navigate to the infrastructure folder and execute the following command to generate the Service Fabric ClusterConfig.json file.

   `.
   \New-SFClusterConfig.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -TemplateConfig <ServiceFabricStandaloneInstallerPath>\ClusterConfig.XS09.MultiMachine.json`

4. Additional modifications to your cluster configuration may be necessary based on your environment. For more information, see, Step 1B: Create a multi-machine cluster, Secure a standalone cluster on Windows using X.509 certificates, and Create a standalone cluster running on Windows Server.

5. Copy the generated ClusterConfig.json file to the <ServiceFabricStandaloneInstallerPath>.

6. Navigate to the <ServiceFabricStandaloneInstallerPath> in Windows PowerShell by using elevated privileges. Run the following command to test ClusterConfig.

   `.
   \Configure-PreReqs.ps1`
7. If the test is successful, run the following command to deploy the cluster.

```powershell
./TestConfiguration.ps1 -ClusterConfigFilePath .\clusterConfig.json
```

# If using offline (internet-disconnected) install
# ./CreateServiceFabricCluster.ps1 -ClusterConfigFilePath .\ClusterConfig.json -
# FabricRuntimePackagePath <Path to MicrosoftAzureServiceFabric.cab download>

```powershell
./CreateServiceFabricCluster.ps1 -ClusterConfigFilePath .\ClusterConfig.json
```

8. After the cluster is created, open the Service Fabric explorer on any client machine to validate the installation.

   a. Install the Service Fabric client certificate in CurrentUser\My if it isn’t already installed.
   b. Go to IE settings > Compatibility Mode, and clear the Display Intranet sites in compatibility mode check box.
   c. Go to https://sf.d365ffo.onprem.contoso.com:19080, where sf.d365ffo.onprem.contoso.com is the host name of the Service Fabric cluster that is specified in the zone. If DNS name resolution isn’t configured, use the IP address of the machine.
   d. Select the client certificate. The Service Fabric explorer page appears.
   e. Verify that all nodes appear as green.

   **IMPORTANT**

   If your client machine is a server machine like Windows Server 2016, you must turn off the IE Enhanced Security Configuration when you access the Service Fabric explorer page. If any antivirus software is installed, ensure you set exclusion following the guidance in the Service Fabric documentation.

11. Configure LCS connectivity for the tenant

Deployment and servicing of Finance + Operations is orchestrated through LCS by using an on-premises local agent. To establish connectivity from LCS to the Finance + Operations tenant, you must configure a certificate that enables the local agent to act on behalf on your Azure AD tenant (for example, Contoso.onmicrosoft.com).

Use the on-premises agent certificate that you acquired from a certificate authority or the self-signed certificate that you generated by using scripts.

The on-premises agent certificate can be reused across multiple sandbox and production environments per tenant.

Only user accounts that have the Global Administrator directory role can add certificates to authorize LCS. By default, the person who signs up for Microsoft 365 for your organization is the global administrator for the directory.

   **IMPORTANT**

   - You must configure the certificate exactly one time per tenant. All on-premises environments under the same tenant must use the same certificate to connect with LCS.
   - If you run this in a server machine like Windows Server 2016, you must turn off the IE Enhanced Security Configuration temporarily. If you don’t, the Azure login window content will be blocked.

1. Sign in to the customer’s Azure portal to verify that you have the Global Administrator directory role.

2. Determine whether the certificate is already registered by running the following script from the
### 2. Set up file storage

You must set up the following SMB 3.0 file shares:

- A file share that stores user documents that are uploaded to AOS (for example, `\DAX7SQLAOFILE1\aos-storage`).
- A file share that stores the latest build and configuration files to orchestrate the deployment (for example, `\DAX7SQLAOFILE1\agent`).

**WARNING**

Keep this file share path as short as possible to avoid exceeding the maximum path length on the files that will be put in the share.

For information about how to enable SMB 3.0, see [SMB Security Enhancements](#).
1. On the file share machine, run the following command.

   ```bash
   Install-WindowsFeature -Name FS-FileServer -IncludeAllSubFeature -IncludeManagementTools
   ```

2. Follow these steps to set up the `\DAX7SQLAOFILE1\aos-storage` file share:
   a. In Server Manager, select **File and Storage Services > Shares**.
   b. Select **Tasks > New Share** to create a new share. Name the share `aos-storage`.
   c. Leave **Allow caching of share** selected.
   d. Check **Encrypt data access**.
   e. Grant **Modify** permissions for every machine in the Service Fabric cluster except OrchestratorType.
   f. Grant **Modify** permissions for the user AOS domain user (contoso\AXServiceUser) and the gMSA user (contoso\svc-AXSF$).

   **NOTE**
   You may need to enable **Computers** under **Object Types** to add machines or enable **Service Accounts** under **Object Types** to add service accounts.

3. Follow these steps to set up the `\DAX7SQLAOFILE1\agent` file share:
   a. In Server Manager, select **File and Storage Services > Shares**.
   b. Select **Tasks > New Share** to create a new share. Name the share `agent`.
   c. Grant **Full-Control** permissions to the gMSA user for the local deployment agent (contoso\svc-LocalAgent$).
13. Set up SQL Server

1. Install SQL Server 2016 SP2 with high availability. (Unless you're deploying in a sandbox environment, where one instance of SQL Server is sufficient. You may want to install SQL Server with high availability in sandbox environments to test high-availability scenarios.)

   **IMPORTANT**
   You must enable the SQL Server and Windows Authentication mode.

   You can install SQL Server with high availability either as SQL clusters that include a Storage Area Network (SAN) or in an Always-On configuration. Verify that the Database Engine, SSRS, Full-Text Search, and Management Tools are already installed.

   **NOTE**
   Make sure that Always-On is set up as described in Select Initial Data Synchronization Page (Always On Availability Group Wizards), and follow the instructions in To Prepare Secondary Databases Manually.

2. Run the SQL service as a domain user or a group-managed service account.

3. Get an SSL certificate from a certificate authority to configure SQL Server for Finance + Operations. For testing purposes, you can create and use a self-signed certificate or an AD CS certificate. You will need to replace the computer name and domain name in the following examples.

   **Certificates for an Always-On SQL instance**

   If you are setting up testing certificates for Always-On, use the following **remoting** script. This will perform the same as the following **manual** script and steps a-e.

   a. Self-signed certificate
14. Configure the databases

1. Sign in to LCS.

**IMPORTANT**

If remoting was used, be sure to execute the clean up steps when the setup is complete. See the 20. Tear down CredSSP section for more information.

---

**Manual self-signed steps for an Always-On SQL instance or Windows Server Failover Clustering with SQL Server**

For each node of the SQL cluster, follow these steps.

**a.** Run the following PowerShell script on each of the SQL Server Always-On replicas.

```
# Manually create certificate for each SQL Node (i.e. 2 nodes = 2 certificates)
# Run script on each node
$computerName = $env:COMPUTERNAME.ToLower()
$domain = $env:USERDNSDOMAIN.ToLower()
$listenerName = 'dax7sqlaosqla'
$cert = New-SelfSignedCertificate -Subject "$computerName.$domain" -DnsName "$listenerName.$domain", $listenerName, $computerName -Provider 'Microsoft Enhanced RSA and AES Cryptographic Provider' -CertStoreLocation "cert:\LocalMachine\My" -KeyAlgorithm "RSA" -HashAlgorithm "sha256" -KeyLength 2048
```

**b.** Grant certificate permissions to the account that is used to run the SQL service.

   - a. Open Manage Computer Certificates (certlm.msc).
   - b. Right-click the certificate created, and then select **Tasks > Manage Private Keys**.
   - c. Add in the SQL Server service account and grant Read access.

**c.** Enable **ForceEncryption** and the new **Certificate** in Microsoft SQL Server Configuration Manager.

   - a. In **SQL Server Configuration Manager**, expand **SQL Server Network Configuration**, right-click **Protocols for [server instance]**, and then select **Properties**.
   - b. In the **Properties** dialog box, on the **Certificate** tab, select the desired certificate from the drop-down menu for the **Certificate** box.
   - c. In the **Properties** dialog box, on the **Flags** tab, in the **ForceEncryption** box, select **Yes**.
   - d. Select **OK** to save.

**d.** Export the certificate (.cer file) from each SQL cluster node, and install it in the trusted root of each Service Fabric node. You will have a minimum of 2 certificates for the Always-On cluster, but there may be more if you have additional replicas.

**e.** Restart the SQL Server service.

**NOTE**

For more information, see How to enable SSL encryption for an instance of SQL Server by using Microsoft Management Console.
2. On the dashboard, select the **Shared asset library** tile.

3. On the **Model** tab, select the demo data for the release that you want and download the zip file.

<table>
<thead>
<tr>
<th>RELEASE</th>
<th>DEMO DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>On-premises General Availability (GA) release</td>
<td>Dynamics 365 for Operations on-premises - Demo data</td>
</tr>
</tbody>
</table>

4. The zip file contains empty and demo data .bak files. Select the .bak file, based on your requirements. For example, if you require demo data, download the AxBootstrapDB_Demodata.bak file.

5. Ensure the database section in the infrastructure\ConfigTempate.xml is configured correctly with the following:
   a. The database name.
   b. The db file and log settings. The db settings should not be lower than the defaults specified.
   c. The path to the backup file downloaded from LCS Shared Asset library. The default name for the Finance + Operations database is AXDB.

   **WARNING**
   - The user running the SQL service and the user running the scripts should have READ access on the folder or share where the backup file is located.
   - If a database with the same name exists, the database will be reused.

6. Copy the **infrastructure** folder to the SQL Server machine and navigate to it in a PowerShell window with elevate privileges.

   **Configure the OrchestratorData database**
   1. Execute the following script.

      ```powershell
      .\Initialize-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName Orchestrator
      ```

      The script will do the following:
      - Create an empty database named **OrchestratorData**. This database is used by the on-premises local agent to orchestrate deployments.
      - Grant the local agent gMSA (svc-LocalAgent$) **db_owner** permissions on the database.

   **Configure the Finance + Operations database**
   1. Execute the following scripts.

      ```powershell
      .\Initialize-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName AOS
      .\Configure-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName AOS
      ```

      The **Initialize-Database.ps1** script will do the following:
      a. Restore the database from the specified backup file.
b. Create a new user that has SQL authentication enabled (axdbadmin).

c. Map users to database roles based on the following table for AXDB.

<table>
<thead>
<tr>
<th>USER</th>
<th>TYPE</th>
<th>DATABASE ROLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>svc-AXSF$</td>
<td>gMSA</td>
<td>db_owner</td>
</tr>
<tr>
<td>svc-LocalAgent$</td>
<td>gMSA</td>
<td>db_owner</td>
</tr>
<tr>
<td>svc-FRPS$</td>
<td>gMSA</td>
<td>db_owner</td>
</tr>
<tr>
<td>svc-FRAS$</td>
<td>gMSA</td>
<td>db_owner</td>
</tr>
<tr>
<td>axdbadmin</td>
<td>SqlUser</td>
<td>db_owner</td>
</tr>
</tbody>
</table>

Map users to database roles based on the following table for TempDB.

<table>
<thead>
<tr>
<th>USER</th>
<th>TYPE</th>
<th>DATABASE ROLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>svc-AXSF$</td>
<td>gMSA</td>
<td>db_datareader, db_datawriter, db_ddladmin</td>
</tr>
<tr>
<td>axdbadmin</td>
<td>SqlUser</td>
<td>db_datareader, db_datawriter, db_ddladmin</td>
</tr>
</tbody>
</table>

The `Configure-Database.ps1` script will do the following:

a. Set READ_COMMITTED_SNAPSHOT ON

b. Set ALLOW_SNAPSHOT_ISOLATION ON

c. Set the specified database file and log settings

d. GRANT VIEW SERVER STATE TO axdbadmin

e. GRANT ALTER ANY EVENT SESSION TO axdbadmin

f. GRANT VIEW SERVER STATE TO [contoso]\svc-AXSF$

g. GRANT ALTER ANY EVENT SESSION TO [contoso]\svc-AXSF$

2. Run the following command to reset the database users.

```
.
Reset-DatabaseUsers.ps1 -DatabaseServer '<FQDN of the SQL server>' -DatabaseName '<AX database name>'
```

**Configure the Financial Reporting database**

1. Execute the following script.

```
.
Initialize-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName MR
```

The script will do the following:

a. Create an empty database named **FinancialReporting**.

b. Map the users to database roles based on the following table.
15. **Encrypt credentials**

1. On any client machine, install the encryption certificate in the LocalMachine\My certificate store.

2. Grant the current user read access to the private key of this certificate.

3. Create the Credentials.json file, as shown here.

```json
{
   "AosPrincipal": {
      "AccountPassword": "<encryptedDomainUserPassword>"
   },
   "AosSqlAuth": {
      "SqlUser": "<encryptedSqlUser>",
      "SqlPwd": "<encryptedSqlPassword>"
   }
}
```

- **AccountPassword** is the encrypted domain user password for the AOS domain user (contoso\axserviceuser).
- **SqlUser** is the encrypted SQL user (axdbadmin) that has access to the Finance + Operations database (AXDB), and **SqlPassword** is the encrypted SQL password.

4. Copy the .json file to the SMB file share, \AX7SQLAOFILE1\agent\Credentials\Credentials.json.

5. Update the Credentials.json file with encrypted values.

```powershell
# Service fabric API to encrypt text and copy it to the clipboard.
Invoke-ServiceFabricEncryptText -Text '<textToEncrypt>' -CertThumbprint '<DataEncipherment Thumbprint>' -CertStore -StoreLocation LocalMachine -StoreName My | Set-Clipboard
```

**IMPORTANT**

Before you can invoke `Invoke-ServiceFabricEncryptText`, you need to install Microsoft Azure Service Fabric SDK. If you encounter the following error, "Invoke-ServiceFabricEncryptText is not recognized command" after you install the Azure Service Fabric SDK, restart the computer and retry.

**WARNING**

After you've finished invoking all `Invoke-ServiceFabricEncryptText` commands, remember to delete the Windows PowerShell history. Otherwise, your non-encrypted credentials will be visible.

16. **Set up SSIS**

To enable Data management and Integration workloads, SSIS must be installed on each of the AOS virtual machines. Complete the following steps on each AOS virtual machine.

1. Verify that the machine has access to the SSIS installation and open the SSIS Setup Wizard.
17. Set up SSRS

2. In the **Feature Selection** window, in the **Features** pane, select the **Integration Services** and **SQL Client Connectivity SDK** check boxes.

3. Complete the setup and verify that the installation was successful.

For more information, see Install integration services.

**18. Configure AD FS**

Before you can complete this procedure, AD FS must be deployed on Windows Server 2016. For information about how to deploy AD FS, see Deployment Guide Windows Server 2016 and 2012 R2 AD FS Deployment Guide.

Finance + Operations requires additional configuration beyond the default out-of-box configuration of AD FS. The following Windows PowerShell commands must be run on the machine where the AD FS role service is installed. The user account must have enough permissions to administer AD FS. For example, the user must have a domain administrator account. For complex AD FS scenarios, consult your domain administrator.

1. **Configure the AD FS identifier so that it matches the AD FS token issuer.**

   This command is related to adding new users using the **Import users** option on the **Users** page (System administration > Users > Users) in the Finance + Operations client.

   ```powershell
   $adfsProperties = Get-AdfsProperties
   Set-AdfsProperties -Identifier $adfsProperties.IdTokenIssuer
   ```

2. You should disable Windows Integrated Authentication (WIA) for intranet authentication connections, unless you've configured AD FS for mixed environments. For more information about how to configure WIA so that it can be used with AD FS, see Configure browsers to use Windows Integrated Authentication (WIA) with AD FS.

   This command is related to using forms authentication upon signing into the Finance + Operations client. Other options, such as single sign-on, are not supported.

   ```powershell
   Set-AdfsGlobalAuthenticationPolicy -PrimaryIntranetAuthenticationProvider FormsAuthentication, MicrosoftPassportAuthentication
   ```

3. For sign-in, the user's email address must be an acceptable authentication input.

   This command is related to setting up email claims. Other options, such as transformation rules, may be available which require additional setup.

   ```powershell
   Add-Type -AssemblyName System.Net
   $fqdn = ([System.Net.Dns]::GetHostEntry('localhost').HostName).ToLower()
   $domainName = $fqdn.Substring($fqdn.IndexOf('.')+1)
   Set-AdfsClaimsProviderTrust -TargetIdentifier 'AD AUTHORITY' -AlternateLoginID mail -LookupForests $domainName
   ```

In order for AD FS to trust Finance + Operations for the exchange of authentication, various application entries...
must be registered in AD FS under an AD FS application group. To speed up the setup process and help reduce errors, you can use the following script for registration. Copy the Publish-ADFSApplicationGroup.ps1 script and D365FO-OP directory to a machine where the AD FS role service is installed. Then run the script by using a user account that has enough permissions to administer AD FS. (For example, use an administrator account.)

For more information about how to use the script, see the documentation that is listed in the script. Make a note of the client IDs that are specified in the output, because you will need this information in LCS in a later step. Should you lose the client IDs, log in to the machine which has AD FS installed, open Server Manager > Tools > AD FS Management > Application Groups > Microsoft Dynamics 365 for Operations On-premises and find the client IDs under the native applications.

NOTE
If you want to reuse your previously configured AD FS server for additional environments, see Reuse the same AD FS instance for multiple environments.

```powershell
.\Publish-ADFSApplicationGroup.ps1 -HostUrl 'https://ax.d365ffo.onprem.contoso.com'
```

Finally, make sure that you can access the AD FS OpenID Configuration URL on a Service Fabric node of the AOSNodeType type. To perform this check, try to open

https://<adfs-dns-name>/adfs/.well-known/openid-configuration

in a web browser. If you receive a message that states that the site isn't secure, you haven't added your AD FS SSL certificate to the Trusted Root Certification Authorities store. This step is described in the AD FS deployment guide, and if you are using remoting, you can use the following script to install the certificate on all nodes in the Service Fabric cluster:

```powershell
.\Install-ADFSCert-AllVMs.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

If you successfully access the URL, a JavaScript Object Notation (JSON) file is returned that contains your AD FS configuration, and you will see that your AD FS URL is trusted.

You've now completed the setup of the infrastructure. The following sections describe how to navigate to LCS to
19. Configure a connector and install an on-premises local agent

1. Sign in to LCS, and open the on-premises implementation project.

2. On the hamburger menu, select Project settings.

3. Select On-premises connectors.

4. Select Add to create a new connector.

5. On the Setup host infrastructure tab, download the agent installer.

6. Verify that the zip file is unblocked. Right-click the file, and then select Properties. In the dialog box, select Unblock.

7. Unzip the agent installer on one of the Service Fabric nodes of the OrchestratorType type.

8. On the Configure agent tab, enter the configuration settings. Execute the following script on any machine with access to it and the configuration file, to get the needed values.

   ```
   .\Get-AgentConfiguration.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
   ```

9. Save the configuration, and then select Download configurations to download the localagent-config.json configuration file.
10. Copy the localagent-config.json file to the machine where the agent installer package is located.

11. In a Command Prompt window, run the following command by navigating to the folder that contains the agent installer.

   ```bash
   LocalAgentCLI.exe Install <path of config.json>
   ```

   **NOTE**
   
   The user who runs this command must have `db_owner` permissions on the OrchestratorData database.

12. After the local agent is successfully installed, navigate back to your on-premises connector in LCS.

13. On the Validate setup tab, select Message agent to test for LCS connectivity to your local agent. When a connection is successfully established, the page will resemble the following illustration.

   ssk-onprem1

20. **Tear down CredSSP, if remoting was used**

   If any of the remoting scripts were used during setup, be sure to execute the following script when there are breaks in the setup process, or the setup has finished.

   ```powershell
   \Disable-CredSSP-AllVMs.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
   ```

   If the previous remoting PowerShell window was accidentally closed and CredSSP was left enabled, the script
21. Deploy your Finance + Operations environment from LCS

1. In LCS, navigate to your on-premises project, go to Environment > Sandbox, and then select Configure. Execute the following script on the primary domain controller VM, which must have access to ADFS and the DNS server settings, to get the needed values.

   ```plaintext
   .\Get-DeploymentSettings.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
   ```

2. For new deployments, select your environment topology, and then complete the wizard to start your deployment.

3. If you have an existing Platform update 8 or Platform update 11 deployment:
   - Update the local agent. See Update the local agent for more details.
   - Validate the local agent from LCS.
   - Deploy Platform update 12 while going through the steps in Reconfigure environments to take a new platform or topology.

4. LCS will assemble the Service Fabric application packages for your environment during the preparation phase. It then sends a message to the local agent to start deployment. You will notice the Preparing status as below.

   ![Preparing status](image)

   Click Full details to take you to the environment details page, as shown below.
5. The local agent will now pick up the deployment request, start the deployment, and communicate back to LCS when the environment is ready. When deployment starts, the status will change to **Deploying**, as shown.

If the deployment fails, the **Reconfigure** button will become available for your environment in LCS, as shown below. Fix the underlying issue, click **Reconfigure**, update any configuration changes, and click **Deploy** to retry the deployment.
See the Reconfigure environments to take a new platform or topology topic for details about how to reconfigure. The following graphic shows a successful deployment.

22. Connect to your Finance + Operations environment

In your browser, navigate to https://[yourD365FOdomain]/namespaces/AXSF, where yourD365FOdomain is the domain name that you defined in the Plan your domain name and DNS zones section of this topic.

Additional resources

- Apply updates to on-premises deployments
- Redeploy on-premises environments
- Configure document management
- Import Electronic reporting (ER) configurations
- Document generation, publishing, and printing in on-premises deployments
- Configure proxies for on-premises environments
- Set up technical support for Finance and Operations apps
- Client internet connectivity
Known issues

Error "Key does not exist" when running the New-D365FOGMSAAccounts cmdlet
If this is your first time creating and generating group Managed Service Account passwords in your domain, you need to first create the **Key Distribution Services KDS Root Key**. For more information, see [Create the Key Distribution Services KDS Root Key](#).

Error "The WinRM client cannot process the request" when running the remoting script Configure-Prereqs-AllVms cmdlet
You need to follow the instructions in the error message to enable the computer policy **Allow delegation fresh credentials** in all machines of Service Fabric cluster.

Error "Not process argument transformation on parameter 'Test'. Cannot convert value "System.String" to type "System.Management.Automation.SwitchParameter" when running the Config-Prereqs-AllVms cmdlet
To work around this error, remove "-Test:$Test" in line 56 of Config-Prereqs-AllVms.ps1, which is found under the **Infrastructure** folder.

Error "Not process argument transformation on parameter 'Test'. Cannot convert value "System.String" to type "System.Management.Automation.SwitchParameter" when running the Complete-Prereqs-AllVms cmdlet
To work around this error, remove "-Test:$Test" in line 56, 61 and 66 of Complete-Prereqs-AllVms.ps1 which is found under the **Infrastructure** folder.

Error "Install-WindowsFeature: The request to add or remove features on the specified server failed" when running Configure-Prereqs on MRType and ReportServerTyoe servers
.NET Framework 3.5 is required in MRType and ReportServerType servers. By default, however, .NET Framework 3.5 source files aren’t included in your Windows Server 2016 installation. To work around this error, install it and specify the source files using the **source** option when you manually add new features by server manager.

Error "MSIS7628: Scope names should be a valid Scope description name in AD FS configuration" when running the Publish-ADFSApplicationGroup cmdlet
This error occurs because of an OpenID scope **allatclaims** that is required by the D365FO-OP-ADFSApplicationGroup, but it might be missing in some Windows Server 2016 installation. To work around this error, add the scope description **allatclaims** through AD FS Management\Service\Scope Descriptions.

Error "ADMIN0077: Access control policy does not exist: Permit everyone" when running the Publish-ADFSApplicationGroup cmdlet
When your AD FS is installed with a non-English version of Windows Server 2016, the permit everyone access control policy is created with your local language. Invoke the cmdlet by specifying **AccessControlPolicyName** parameter as:

```
```
This topic covers the installation steps for Commerce channel components in an on-premises environment.

**IMPORTANT**

There is currently a known issue where self-service packages will not correctly apply to on-premises environments. For that reason it is recommended to pull the installers directly from Microsoft Dynamics Lifecycle Services (LCS) and use them as needed. Commerce headquarters would thereby no longer be used to download the installers but only the configuration files as needed.

Channel functionality, in an on-premises environment, is enabled exclusively via use of Commerce Scale Unit (self-hosted). For an overview, see Commerce Scale Unit (self-hosted).

Unlike a cloud deployment, an on-premises environment does not enable seamless, high-availability deployment of channel components via Lifecycle Services (LCS). The only way to use channel components is by installing Commerce Scale Unit (self-hosted).

**Prerequisites**

Before you can start installation of channel components, you must first complete all prior installation steps for an on-premises environment. These steps are listed in Set up and deploy on-premises environments (Platform update 12 and later). In addition, version 8.1.1 must be installed in order for Commerce have full functionality. We recommend that you update to version 8.1.2.

**NOTE**

It is critical to ensure that a secure network, that is not publicly accessible, is used to connect Commerce Scale Unit to Headquarters. You must also restrict network access to Headquarters, so access is only allowed to known Commerce Scale Unit devices via network filtering or other means. This means that a firewall must exist and using a safe list is highly recommended.

**Installation steps**

1. On the previously created Application share, (not the LocalAgent share folder), create a new folder called **selfservicepackages** in the root directory of the share location.

2. On each AOS computer, create an easily accessible directory, such as **C:/selfservicepackages**.

3. On one AOS computer (which one does not matter), run the following PowerShell script.

```
./RetailUpdateDatabase.ps1 -envName '<Environment name>' -AosUrl 'https://<My Environment Name>.com/namespaces/AXSF/' -SendProductSupportTelemetryToMicrosoft
```
4. On each AOS computer, run the following PowerShell script.

```powershell
./RetailUpdateDatabase.ps1 -RetailSelfServicePackages 'C:\RetailSelfService\Packages'
```

**NOTE**
The parameter `-RetailSelfServicePackages` is the full path location created in the beginning of this step (`C:/selfservicepackages`).

5. Download the appropriate binary update from LCS to have the Commerce installers. For instructions, see [Download updates from Lifecycle Services (LCS)](download_updates_from_lcs).

6. Extract the zip file and copy all self-service installers into the folder `C:/selfservicepackages` defined and created in step 2 in each of the AOS machines. The six self-service installers include:

- AsyncServerConnectorServiceSetup.exe
- RealtimeServiceAX63Setup.exe
- HardwareStationSetup.exe
- ModernPosSetup.exe
- ModernPosSetupOffline.exe
- StoreSystemSetup.exe

**NOTE**
Cloud environments can synchronize self-service installers through Headquarters from what is available in LCS ([Synchronize self-service installers in Dynamics 365 Commerce](synchronize_self_service_installers_in_dynamics_365_commerce)). On-premises environments cannot utilize this functionality, however, these environments can still download from LCS. The SDK is available in the deployable package zip file. The self-service installers are available from the LCS Asset library. You can utilize the upload and download mechanism from within LCS, but the Headquarters synchronization functionality will not work.

7. Navigate to the AD FS machine, then go to the InfrastructureScripts folder. This is the same file directory...
where the previously run PowerShell script was located (RetailUpdateDatabase.ps1). Find the PowerShell script Create-ADFSServerApplicationForRetail.ps1.

8. On the AD FS machine that you’re currently using, run this script in a new PowerShell window using the command `\Create-ADFSServerApplicationForRetail` -HostUrl 'https://ax.d365ffo.onprem.contoso.com', where the HostUrl value can be found in Service Fabric. To find the HostUrl value, go to Service Fabric > Application fabric/AXSF > Details > Aad_AADValidAudience.

9. Access the newly generated Server application from the Application Groups in AD FS Management.

10. Edit the newly generated Server application and select Reset the Secret.

   **NOTE**
   It is an important security measure to run this script for each Commerce Scale Unit. This maximizes security and minimizes the workload in case of a security breach.

   It is critical to keep this secret safe. This secret should only be copied once and never stored on the system. The Client ID and Secret generated will be used during the Commerce Scale Unit installer, so it is required to be used at a later time. You can always reset the secret again, but it must then be updated on any Commerce Scale Unit that used the previous secret.

11. Go to Retail and Commerce > Headquarters setup > Commerce scheduler > Connector for Microsoft Dynamics AX.


13. In the Profile field, enter the value Default. If needed, enter a description in the Description field.

   **NOTE**
   It is possible for the following fields in steps 12 through 14 to already have values. If this occurs, skip those steps and continue from there. What is important is to have a selectable profile title (default in this case).

14. In the Web application name field, enter RetailCDXRealTimeService.

15. In the Protocol field, select https.

16. In the Common name field, enter AXServiceUser@contoso.com.

17. Select Save on the Action Pane.

18. In Headquarters, go to Retail and Commerce > Headquarters setup > Parameters > Commerce shared parameters.


20. Under the sub-heading Transaction service legacy properties, select the Real-time Service profile field, and then select the newly created Default value.

21. Select the Identity providers tab.

22. On the Identity providers FastTab, select Add.

23. In the new Issuer row, enter the new Identity provider value https://sts.windows.net/ in the field.


25. Go to Retail and Commerce > Headquarters setup > Parameters > Commerce parameters.
26. On the **General** tab, select the **Initialize** link to configure seed data for Commerce functionality.

**NOTE**

- Read the important message at the beginning of this article regarding a known issue with installers no longer functioning through headquarters for download.
- The installers will not download from their relevant pages the first time a download is attempted. This is because the installers have only just been placed into the download location and the associated database values do not yet exist. In Headquarters, when the **Download** functionality is attempted (for example, Commerce Scale Unit or Modern POS), an error will display and then an automated upload functionality will be initiated to allow the installers to be downloaded the second time that the download is attempted. (Wait one minute before attempting to download the installer again).
- The Peripheral Simulator (downloaded on the Hardware profile page in headquarters) will not be available until at least one Hardware profile has been created and is functional. After that point has been achieved, the following script can be run.

```bash
.RetailUpdateDatabase.ps1 -envName 'LBEnv1' -UpdateRetailHardwareProfileSelfServicePackage
```

27. Follow the installation steps for installing the Commerce Scale Unit. For instructions, see **Configure and install Commerce Scale Unit (self-hosted)**. At multiple locations in this document there will be notes referencing changes to the instructions for an on-premises deployment. It is important to note each of these changes.
This topic describes how to develop customizations and extensions, and deploy them to an on-premises environment. On-premises environments are also referred to as local business data (LBD) environments. This topic focuses on the ways that this process differs from the process in a run-time cloud environment.

The process has the following main steps:

1. Deploy your development and build environments.
2. Create a deployable package of your code and customizations.
3. Upload the deployable package to your project in Microsoft Dynamics Lifecycle Services (LCS).
4. Configure and deploy an on-premises runtime environment that includes your deployable package. This environment can be either a sandbox environment or a production environment.

The following sections provide more information about this process.

**Development tools and platform**

Whether you’re developing, extending, or customizing cloud applications or on-premises applications, the development platform, tools, and environments (virtual machines [VMs]) are the same. Your custom code is developed on the same development VMs, regardless of whether your target runtime environments are in a cloud environment or an on-premises environment.

For detailed information about development, see the [Develop and customize home page](#). For information about extensibility and customization, see the [Extensibility home page](#). For information about building, testing, and continuous delivery, see the [Continuous delivery home page](#).

**Deploy development and build environments**

You can use an on-premises LCS project to deploy build and development environments on Microsoft Azure by using your own Azure subscription. Alternatively, you can download a virtual hard disk (VHD) for local development.

To deploy a development or build environment in your Azure subscription, or to download a development VHD, open the [Cloud-hosted environments](#) page in LCS.
Then follow these steps.

1. Click **Add**.

   **Cloud-hosted environments**

   - **Environment name**
   - **Topology type**
   - **Dephi**

2. Select **Azure** or **Locally**. If you select **Locally**, find and download a development VHD. If you select **Azure**, you're prompted to select one of three topologies: **Build and Test**, **Demo**, or **Development**.

3. Complete the deployment steps, and deploy a VM in your Azure subscription.

For more information about how to configure a local development VHD, see [Deploy and access development environments](#).
Create and upload a deployable package to the LCS Asset library

When you complete a phase of development, and are ready to deploy your code to a sandbox or production on-premises environment, you must create an application deployable package from your models. This process doesn’t differ from the process for cloud environments.

If you’re using automated builds (a build environment), the build process creates an application deployable package for you. You can also create an application deployable package from Microsoft Visual Studio in your development environment. For more information on how to create an application deployable package in your development environment, see Create deployable packages of models.

When your deployable package is ready, follow these steps to upload it to your LCS project's Asset library.

1. Open the **Asset library** page.

2. Click the **Software deployable package** tab.
3. Click the plus sign (+) to upload the deployable package.

Configure an on-premises runtime environment that uses your code

As of the July 2017 release of Microsoft Dynamics 365 for Finance and Operations (on-premises), you can apply your customizations and extensions only during the deployment of a sandbox or production environment.

1. In your LCS project, click **Configure** to deploy your environment.

2. In the deployment tool, when you must enter the environment name, click **Advanced settings**.

   **On premise topology**

   Dynamics 365 for Finance and Operations - OnPremise (July 2017 update):

   - Environment name: Sandbox1
   - Connector: test

   By selecting this check box, you agree to the pricing and licensing terms:

   - Microsoft Dynamics Software License Terms
   - Microsoft Online Services Privacy Statement

3. Click the **Customize solution assets** tab.
4. In the **Select the AOT packages to be deployed** field, select the application (AOT) deployable package that contains your customizations. This field lists all the AOT packages in your Asset library.

5. Click **Done** to close the **Deployment settings** page, and then continue with the environment deployment process.
This topic explains how to apply supported updates to Dynamics 365 Finance + Operations (on-premises). All updates to on-premises environments are done through Microsoft Dynamics Lifecycle Services (LCS).

Search for and download updates

For more information about how to find the updates that you can apply to your on-premises environment, see Issue search in Lifecycle Services (LCS). For information about how to download updates from the tiles in the Updates section of the Environment details page in LCS, see Download updates from Lifecycle Services (LCS).

NOTE

When you are updating an on-premises environment, always select updates from the update tiles on the Environment details page. If you select updates from another location, the updates might not work.

Update an on-premises deployment

You can apply updates to an on-premises environment either during deployment or after the deployment is completed.

While an on-premises environment is being deployed, you can select to deploy a custom package in the Advanced settings. For more information about how to apply customizations or application X++ updates, see Develop and deploy custom models to on-premises environments.

To apply updates to an on-premises environment after it has been deployed, in LCS, on the Environment details page for the environment, under Maintain, select Apply updates.

NOTE

You can apply updates after deployment only on environments that have Platform update 12 for Finance and Operations or later. The environment must also have the latest version of the local agent available in LCS. For more information, see Update the local agent. If you’re on a platform version that is older than Platform update 12, you can reconfigure an environment that is already deployed to update the customizations or update to the latest platform release. For more information about how to redeploy an environment, see Redeploy on-premises environments.

Apply application or binary updates through LCS

The following steps can be used to apply X++, All Binary, or Platform binary updates.

IMPORTANT

The application of updates requires downtime for your environment. Therefore, no business transactions can be performed in the environment during the update. When you complete the following steps, verify that the system isn't being used, and that an official downtime notice has been communicated to all system users.
Prerequisites

- Before you begin, complete a full backup of the Management Reporter (MR), Microsoft Dynamics AX, and Microsoft SQL Server Reporting Services (SSRS databases). Although the code is restored through LCS, the database must be manually restored to help guarantee that there is no data loss.

- Update your environment to the latest build of Platform update 12.

- Update the local agent to the latest version. For more information, see Update the local agent.

- Depending on the type of update, complete the following steps to generate a deployable package:
  - **Platform binary updates** – Download or save the update directly to the Asset library in LCS by following the steps in Download updates from Lifecycle Services (LCS).
  - **Application binary updates** – Download or save the update directly to the Asset library in LCS by following the steps in Download updates from Lifecycle Services (LCS).
  - **Application X++ updates** – Download the required hotfix to your development environment, and then follow the steps in Create deployable packages of models.
  - **Customizations** – Follow the steps in Develop and deploy custom models to on-premises environments.

Update a sandbox environment

1. In the LCS Asset library, upload the deployable package that was generated in the "Prerequisites" section of this topic to the Software deployable packages tab.

2. In LCS, open the on-premises implementation project, and then open the Environment details page of the environment to update.

3. Under Maintain, select Apply updates. A slider shows the updates that were uploaded to the Asset library. Note that only packages that are marked as Valid in the Asset library appear.

If you are on local agent version 2.1.0 and higher, complete the following steps.

1. Select the update, and then click Prepare. Clicking on Prepare will prepare your on-premises environment for servicing.

   **NOTE**
   
   During preparation, the environment state will be Deployed but the Deployment status field will show the progress of Preparation. Steps such formatting the package and downloading the package are executed during preparation. The environment is not directly touched during preparation and hence there is no downtime during the preparation phase. Users can continue to use the system during preparation.

2. After the preparation is complete, you will see Abort and Update Environment buttons. To start applying the update, click Update Environment. If preparation fails, see the "Resolve a failed update application" section later in this topic.

3. In the confirmation message, select Yes. The servicing operation has started on this environment. This is the start of the downtime on your environment.

4. The environment state is changed from Deployed to Deploying.

5. After the update is completed, the environment state is changed back to Deployed. If application of the
Update a production environment

1. Select the update, and then click **Apply**.

2. In the confirmation message, select **Yes**. The servicing operation has started on this environment. This is the start of the downtime on your environment.

3. Environment state changes from **Deployed** to **Preparing**.

   **NOTE**
   During preparation, steps such as formatting the package and downloading the package are executed during preparation. The environment is not directly touched during preparation and hence there is no downtime during the preparation phase. Users can continue to use the system during preparation. However, we recommend that the downtime starts when the environment enters the Preparing state.

4. After preparation is complete, the environment state is changed from **Preparing** to **Deploying**.

5. After the update is completed, the environment state is changed back to **Deployed**. If application of the update fails, the environment state is changed to **Failed**. For information about what to do if package application fails, see the "Resolve a failed update application" section later in this topic.

6. Open the **History** and **Environment details** pages to view the operations that were performed on the environment. You can also view a record of major actions that were performed on the environment, such as deployments, servicing, and rollbacks.

If you are on local agent version lower than 2.1.0, complete the following steps.

1. Select the update, and then click **Apply**.

2. In the confirmation message, select **Yes**. The servicing operation has started on this environment. This is the start of the downtime on your environment.

3. Environment state changes from **Deployed** to **Preparing**.

   **NOTE**
   During preparation, steps such as formatting the package and downloading the package are executed during preparation. The environment is not directly touched during preparation and hence there is no downtime during the preparation phase. Users can continue to use the system during preparation. However, we recommend that the downtime starts when the environment enters the Preparing state.

4. After preparation is complete, the environment state is changed from **Preparing** to **Deploying**.

5. After the update is completed, the environment state is changed back to **Deployed**. If application of the update fails, the environment state is changed to **Failed**. For information about what to do if package application fails, see the "Resolve a failed update application" section later in this topic.

6. Open the **History** and **Environment details** pages to view the operations that were performed on the environment. You can also view a record of major actions that were performed on the environment, such as deployments, servicing, and rollbacks.

**Update a production environment**

Before you update a production environment, you must successfully complete the package application update on a sandbox environment.

1. In the project for the sandbox environment that you applied the package to, open the Asset library, and then, on the **Software deployable packages** tab, select the package, and mark it as a **Release candidate**.

2. On the **Environment details** page, under **Maintain**, select **Apply updates**. In the dialog box, only packages that are marked as a **Release candidate** are shown.

3. Select the Release candidate package to be applied to the Production environment.

4. The rest of the Update flow is the same as that of a sandbox environment. Your update experience will differ based on the version of the local agent running on your environment. We recommend that you always run with the latest version.

**Resolve a failed update application**

When preparation fails, the environment state is **Deployed**. When the application of an update fails, the environment state is **Failed**. The first step is to determine why there is a failure. The location of the logs varies, depending on the stage where the failure occurred:

- **Preparation stage**: If the operation fails during the **Preparation** stage, the logs are uploaded to LCS. In the log files, select **Download logs** to download the log files. If the package has any merge issues, the error is included in the log file.

- **Deploying stage**: If the operation fails during the **Deploying** stage, the logs are located in the on-premises
environment. You must sign in to the environment, and then access the logs and event viewer.

For more information about how to use the troubleshooting logs, see Troubleshoot on-premises deployments.

After you review the logs and determine the cause of the failure, complete one of the following operations to restore the environment to a healthy state. No actions can be performed on an environment that is in a Failed state. The environment must first be restored to a healthy state.

- **Retry failed operation** – If update application fails, select Retry to recover from the failed operation.

- **Abort failed operation** – Because there is no change made to the on-premises environment, if the preparation fails, you have the option to cancel the operation. Select Abort to cancel the preparation.

- **Roll back the update** – To roll back the update that failed, select Rollback. Before you start the rollback, you must restore the database to the last known good state. When you select Rollback, the environment is restored to the last known good state. The environment state is then changed to Preparation, then to Deploying, and then to either Deployed or Failed.

  **NOTE**
  
  The Rollback button doesn't roll back the database. You're responsible for restoring the database to the last known backup that was made before update application. This step is critical to help guarantee that there is no data loss.

- **Refresh the state** – If update application fails during the Preparation stage, the failure is on the LCS side, and update application hasn't yet started. Therefore, the on-premises environment is in a good state. To restore the LCS environment state to Deployed, on the project dashboard page, select Refresh.

- **Delete and redeploy an environment** – If the retry and rollback options don't work, you must delete and redeploy the environment. To delete the environment, on the project dashboard page, select Delete. You then see the option to configure the environment.

  **IMPORTANT**
  
  This option should not be used on a production environment. However, it can be used on a sandbox deployment to restore the environment to a healthy state.

Because this option requires that you do a fresh deployment of the environment, you lose any updates that were previously applied. Any customizations and binary updates must be reapplied to the environment.
You may need to rotate the certificates used by your Dynamics 365 Finance + Operations (on-premises) environment as they approach their expiration date. In this topic, you will learn how to replace the existing certificates and update the references within the environment to use the new certificates.

**WARNING**

The certificate rotation process should be initiated well before the certificates expire. This is very important for the Data Encryption certificate, which could cause data loss for encrypted fields. For more information, see [After certificate rotation](#).

Old certificates must remain in place until the certificate rotation process is complete, removing them in advance will cause the rotation process to fail.

The certificate rotation process should not be carried out on Service Fabric clusters running 7.0.x and 7.1.x. Upgrade your Service Fabric cluster to 7.2.x or later before attempting certificate rotation.

### Preparation steps

1. Rename the original **Infrastructure** folder that you created during the process to [Download setup scripts from LCS](#). Rename the folder to **InfrastructureOld**.

2. Download the latest setup scripts from [Download setup scripts from LCS](#). Unzip the files into a folder that is named **Infrastructure**.

3. Copy **ConfigTemplate.xml** and **ClusterConfig.json** from **InfrastructureOld** to **Infrastructure**.

4. Configure certificates as needed in **ConfigTemplate.xml**. Follow the steps in [Configure certificates](#), specifically these steps.

```powershell
# Create self-signed certs
.
New-SelfSignedCertificates.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

Alternatively, if you have or would like to switch to Active Directory Certificate Services (AD CS) certificates, use this information.

```powershell
# Only run the first command if you have not generated the templates yet.
.
New-ADCSCertificates.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -CreateTemplates
.
New-ADCSCertificates.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

**NOTE**

The AD CS scripts need to run on a domain controller, or a Windows Server computer with Remote Server Admin Tools installed. The AD CS functionality is only available with Infrastructure scripts release 2.7.0 and later.

Self-signed certificates should never be used in production environments. If you're using publicly trusted certificates, manually update the values of those certificates in the ConfigTemplate.xml file.
5. Continue to Setup VMs. The specific steps that are needed for this process include:

a. Export the scripts that must be run on each VM.

```
# Export the script files to be executed on each VM into a directory VMs\<VMName>
.
s Export-Scripts.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

b. Copy the contents of each infrastructure\VMs<VMName> folder into the corresponding VM (if remoting scripts are used, they will automatically copy the content to the target VMs), and then run the following scripts, if they exist. Perform these steps as an Administrator.

```
# If remoting, only execute
# .\Complete-PreReqs-AllVMs.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ForcePushLBDScripts
.
s Import-PfxFiles.ps1
.
s Set-CertificateAcls.ps1
```

c. Run the following script to validate the VM setup.

```
# If remoting, only execute
# .\Test-D365FOConfiguration-AllVMs.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
.
s Test-D365FOConfiguration.ps1
```

6. If axdataenciphermentcert certificates are rotated, you need to regenerate the credentials.json file. For more information, see Encrypt credentials.

7. Run the following PowerShell command to have values that can be used in LCS later. For more information, see Deploy your on-premises environment from LCS.

```
.
s Get-DeploymentSettings.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

---

**Activate new certificates within Service Fabric cluster**

**Service Fabric with certificates that aren't expired**

1. Locate the Clusterconfig.json file for editing. If you cannot find this file, follow steps 2 and 3, otherwise continue to step 4.

2. Run the following command to connect to the Service Fabric Cluster.

```
# Connect to the Service Fabric Cluster from a node within the cluster
Connect-ServiceFabricCluster
```

3. Run the following command to save the configuration file to C\Temp\ClusterConfig.json. (Make sure that the C\Temp path exists.)

```
Get-ServiceFabricClusterConfiguration > C:\Temp\ClusterConfig.json
```
4. Open the **Clusterconfig.json** file for editing and find the following section. If a secondary thumbprint is defined, go to **Clean up old Service Fabric certificates** before you continue.

```
"security": {
    "metadata": "The Credential type X509 indicates this cluster is secured using X509 Certificates. The thumbprint format is - d5 ec 42 3b cb e5 07 fd 83 59 3c 56 b9 d5 31 24 25 42 64.",
    "ClusterCredentialType": "X509",
    "ServerCredentialType": "X509",
    "CertificateInformation": {
        "ClusterCertificate": {
            "X509StoreName": "My",
            "Thumbprint": "Old server thumbprint(Star/SF)*",
            "ThumbprintSecondary": "Old server thumbprint(Star/SF)"
        },
        "ServerCertificate": {
            "X509StoreName": "My",
            "Thumbprint": "Old server thumbprint(Star/SF)*",
            "ThumbprintSecondary": "Old server thumbprint(Star/SF)"
        },
        "ClientCertificateThumbprints": [
            {
                "CertificateThumbprint": "Old client thumbprint",
                "IsAdmin": true
            }
        ]
    }
},
```

5. Replace that section in the file with the following code.

```
"security": {
    "metadata": "The Credential type X509 indicates this cluster is secured using X509 Certificates. The thumbprint format is - d5 ec 42 3b cb e5 07 fd 83 59 3c 56 b9 d5 31 24 25 42 64.",
    "ClusterCredentialType": "X509",
    "ServerCredentialType": "X509",
    "CertificateInformation": {
        "ClusterCertificate": {
            "X509StoreName": "My",
            "Thumbprint": "New server thumbprint(Star/SF)*",
            "ThumbprintSecondary": "Old server thumbprint(Star/SF)"
        },
        "ServerCertificate": {
            "X509StoreName": "My",
            "Thumbprint": "New server thumbprint(Star/SF)*",
            "ThumbprintSecondary": "Old server thumbprint(Star/SF)"
        },
        "ClientCertificateThumbprints": [
            {
                "CertificateThumbprint": "Old client thumbprint",
                "IsAdmin": false
            },
            {
                "CertificateThumbprint": "New client thumbprint",
                "IsAdmin": true
            }
        ]
    }
},
```

6. Edit the new and old thumbprint values.

7. Change clusterConfigurationVersion to the new version, for example 2.0.0.
8. Save the new ClusterConfig.json file.

9. Run the following PowerShell command.

```powershell
# Connect to the Service Fabric Cluster
Connect-ServiceFabricCluster

# Get path of ClusterConfig.json for following command
# Note that after running the following command, you need to manually cancel using the red button
# (Stop Operation) in Windows PowerShell ISE or Ctrl+C in Windows PowerShell, otherwise you will
# receive the following notification, "Start-ServiceFabricClusterConfigurationUpgrade : Operation timed out.". Be aware that the upgrade will proceed.
Start-ServiceFabricClusterConfigurationUpgrade -ClusterConfigPath ClusterConfig.json

# If you are using a single Microsoft SQL Server Reporting Services node, use
UpgradeReplicaSetCheckTimeout to skip PreUpgradeSafetyCheck check, otherwise it will timeout
Update-ServiceFabricClusterUpgrade -UpgradeReplicaSetCheckTimeoutSec 30

# To monitor the status of the upgrade, run the following and note UpgradeState and
UpgradeReplicaSetCheckTimeout
Get-ServiceFabricClusterUpgrade

# While monitoring the status of the upgrade, if UpgradeReplicaSetCheckTimeout was reset to the
default (example 49710.06:28:15), run the following command again
Update-ServiceFabricClusterUpgrade -UpgradeReplicaSetCheckTimeoutSec 30

# When UpgradeState shows RollingForwardCompleted, the upgrade is finished
```

**Service Fabric with or without expired certificates (cluster not accessible)**

Continue this process following Troubleshoot on-premises deployments.

**Update the LocalAgent certificate**

You must reinstall the LocalAgent if:

- You changed the service fabric cluster/server certificate.
- You changed the service fabric client certificate.
- You changed the LocalAgent certificate.

1. Update your current localagent-config.json by replacing the `serverCertThumbprint` and `clientCertThumbprint` values with the new thumbprints.

```json
{
    "serviceFabric": {
        "connectionEndpoint": "192.168.8.22:19000",
        "clusterId": "Orch",
        "certificateSettings": {
            "serverCertThumbprint": "New server thumbprint(Star/SF)",
            "clientCertThumbprint": "New client thumbprint"
        }
    }
}
```

2. Run the following PowerShell command on one of the Orchestrator nodes.
3. Run the following PowerShell command and note the new LocalAgent thumbprint.

\.`LocalAgentCLI.exe Cleanup <path of localagent-config.json>`

```powershell
.\Get-AgentConfiguration.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

4. Follow the steps in Configure LCS connectivity for the tenant.

**NOTE**
If you receive the error *Update to existing credential with KeyId '<key>' is not allowed*, follow the instructions in Error: "Updates to existing credential with KeyId '<key>' is not allowed".

5. Continue with Configure a connector and install an on-premises local agent, specifically the following changes:

- Client certificate thumbprint
- Server certificate thumbprint
- Tenant service principle certificate thumbprint

**IMPORTANT**
Do not create a new connector in LCS. Update the configuration of your existing connector and download the settings again.

---

Update your current deployment configuration

Because you’ve updated your certificates, the configuration file that is present in your environment is outdated and must be manually updated. Otherwise, the clean-up job will probably fail. (This manual update must be done just this one time.)

1. Open your configuration file. You can find the location of this file by running the following command.

```
select Location from DeploymentInstanceArtifact where AssetId='config.json' and DeploymentInstanceId = 'LCSENVIRONMENTID'
```

**NOTE**
Replace `LCSENVIRONMENTID` with the ID of your environment. You can obtain this ID from the page for your environment in LCS.

The beginning of the file should resemble the following example.

```
{
  "serviceFabric": {
    "connectionEndpoint": "192.168.8.22:19000",
    "clusterId": "Orch",
    "certificateSettings": {
      "serverCertThumbprint": "Old server thumbprint(Star/SF)",
      "clientCertThumbprint": "Old client thumbprint"
    }
  },
```
2. Replace the `serverCertThumprint` and `clientCertThumprint` values with the new thumbprints.

```json
{
   "serviceFabric": {
      "connectionEndpoint": "192.168.8.22:19000",
      "clusterId": "Orch",
      "certificateSettings": {
         "serverCertThumbprint": "New server thumbprint(Star/SF)",
         "clientCertThumbprint": "New client thumbprint"
      }
   }
}
```

3. Save and close the file. Remember to close any programs that are accessing this network location. Otherwise, the cleanup process might fail.

**Update deployment settings in LCS**

**NOTE**

Note that the Client, Data Signing, and Encipherment certificates will only be replaced. You will also need to recreate the Credentials.json file, as described in [Encrypt credentials](#).

Before you continue, you need to make a backup of the local Dynamics database.

1. In LCS, select the "Full Details" link for the environment where you want to change the certificates.
2. Select **Maintain** and then select **Update Settings**.

![Maintain Settings](image)

3. Change the thumbprints to the new thumbprints that you previously configured. You can find them in the ConfigTemplate.xml file in the InfrastructureScripts folder.
4. Select **Prepare**.

5. After downloading and preparation is complete, the **Update environment** button will display.

![Deployed]

6. Select **Update environment** to start updating your environment.

7. During the update, the environment will be unavailable.

8. After the environment is successfully updated with the new certificates, you can view the new thumbprints in Service Fabric Cluster Explorer. The names of the thumbprints in Service Fabric Explorer might differ from the names in LCS. However, the values should be the same.

Here is an example of how the name of the same thumbprint might differ.
Update other certificates as needed

1. Always check if the SQL server certificate has expired. For more information, see Set up SQL Server.

2. Check to be sure that the Active Directory Federation Service (ADFS) certificate has not expired.

Clean up old Service Fabric certificates

This procedure should be completed either after a successful certificate rotation or before the next certificate rotation.

1. Remove the old/secondary thumbprints from the cluster configuration. After you've removed them, the appropriate section should resemble the following example.

```json
"security": {
   "metadata": "The Credential type X509 indicates this is cluster is secured using X509 Certificates.
   The thumbprint format is - d5 ec 42 3b 79 cb e5 07 fd 83 59 3c 56 b9 d5 31 24 25 42 64."
   "ClusterCredentialType": "X509",
   "ServerCredentialType": "X509",
   "CertificateInformation": {
      "ClusterCertificate": {
         "X509StoreName": "My",
         "Thumbprint": "server thumbprint(Star/SF)"
      },
      "ServerCertificate": {
         "X509StoreName": "My",
         "Thumbprint": "server thumbprint(Star/SF)"
      },
      "ClientCertificateThumbprints": [
         {
            "CertificateThumbprint": "client thumbprint",
            "IsAdmin": true
         }
      ]
   }
}
```

2. Follow steps 4 through 6 in the Service Fabric with certificates that are not expired section earlier in this topic.

After certificate rotation

Data encryption certificate

This certificate is used to encrypt data stored in the database. By default there are certain fields that are encrypted with this certificate, you can check those fields in Document the values of encrypted fields. However, our API can be used to encrypt other fields that customers deem should be encrypted.

In Platform update 33 and later, the batch job that is named “Encrypted data rotation system job” will use the newly rotated certificate to re-encrypt data. This batch job crawls through your data to re-encrypt all the encrypted data by using the new certificate. It will run for two hours per day until all of the data has been re-encrypted. In order to enable the batch job, a flight and a configuration key need to be enabled. Execute the following commands against your business database (for example, AXDB).
IF (EXISTS(SELECT * FROM SYSFLIGHTING WHERE [FLIGHTNAME] = 'EnableEncryptedDataCrawlerRotationTask'))
  UPDATE SYSFLIGHTING SET [ENABLED] = 1 WHERE [FLIGHTNAME] = 'EnableEncryptedDataCrawlerRotationTask'
ELSE
  INSERT INTO SYSFLIGHTING ([FLIGHTNAME],[ENABLED],[FLIGHTSERVICEID]) VALUES
  ('EnableEncryptedDataCrawlerRotationTask', 1, 0)
ENDIF (EXISTS(SELECT * FROM SECURITYCONFIG WHERE [KEY_] = 'EnableEncryptedDataRotation'))
  UPDATE SECURITYCONFIG SET [VALUE] = 'True' WHERE [KEY_] = 'EnableEncryptedDataRotation'
ELSE
  INSERT INTO SECURITYCONFIG ([KEY_], [VALUE]) VALUES ('EnableEncryptedDataRotation', 'True')
ENDIF

After the above commands have been executed, restart your AOS nodes from Service Fabric Explorer. The AOS will detect the new configuration and will schedule the batch job to run during off hours. After the batch job has been created, the schedule can be modified from the user interface.

WARNING
Make sure that the old Data Encryption certificate is not removed before all encrypted data has been re-encrypted and it has not expired. Otherwise, this could lead to data loss.
The configuration of the local network for a deployment of Dynamics 365 Finance + Operations (on-premises) can affect the features that are available in the web client. In particular, if the network configuration does not allow a client machine to access the internet, several degradations in the web client will occur. These include:

- The Office app launcher and Dynamics 365 areas in the navigation bar will no longer be clickable.
- The Help pane will not be accessible.
- The Ideas portal will not be accessible from the web client.
- Users will see their initials instead of a user image.
- Skype integration will not be available.
- The favorite icon shown in the browser tab will be the browser's default favorite icon instead of the application icon.
- The Open in Excel options are hidden because the Excel Add-in will not run.

In addition to platform features that may not be accessible when the client can't access the internet, there may also be application features that rely on an internet connection that developers will need to hide or turn off. To facilitate this, developers can use the `clientHasRestrictedInternet()` method that has been added to the `Session` class. This method will return true if the client does not have access to the internet.

**Client internet connectivity options**

Client internet connectivity options allow an administrator to manually turn off the external connections that the client makes even when internet connectivity is available. These can be used for troubleshooting issues or to see what the client will look like when internet connectivity is not available. These client internet connectivity options are available out-of-the-box starting in Platform update 16 but are also available in Platform update 15 (via KB 4091764) and Platform update 12 (via KB 4091763).

The client internet connectivity options can be found on the System administration > Setup > Client performance options page.

- **Internet connectivity enabled** - Allows an administrator to turn off all external connections that the web client would otherwise make.
- **Skype presence enabled** - Allows an administrator to turn off external connections to Skype that the web client would otherwise make.

**Why does the client connect to the Skype for Business API when it first loads?**

When the client loads, it performs a quick call (ping) to the Skype for Business API to check if an internet connection is available. If it isn't available then the client functions in a disconnected fashion. An environment doesn’t need to have Skype for Business visible/enabled for this check to be made.
This topic explains how to configure your environment so that you can deploy batch-only and interactive-only Application Object Server (AOS) nodes.

To make this feature available, Microsoft has introduced two new Microsoft Azure Service Fabric node types. For batch-only AOS nodes, the new node type is `BatchOnlyAOSNodeType`. For interactive-only AOS nodes, the new node type is `InteractiveOnlyAOSNodeType`.

The traditional deployment option, where an AOS node is interactive and is running batch jobs, is still supported and isn't affected by these changes.

**Sizing**

For sandbox environments, we recommend that you have at least two nodes of each type.

For production environments, there should be at least three nodes of each type.

**New deployments**

1. When you're describing your configuration as explained in Set up and deploy on-premises environments, edit the `configtemplate.xml` file to add the new node types. When you've finished, the new `NodeType` sections should resemble the following example.
2. Follow the remaining instructions in *Set up and deploy on-premises environments* in the usual way.

**Existing deployments**

1. Follow the instructions in *Remove an AOS node* up through the point where you save the configuration file.

   **IMPORTANT**

   You must use option 1, "Use a configuration file (preferred option)." Do not use option 2.

2. Continue to edit the *ClusterConfig.json* file to add the new node types to the *NodeTypes* section. When you’ve finished, the *NodeTypes* section should resemble the following example.
3. Continue to follow the instructions in Remove an AOS node from the point where you stopped until you’ve finished removing the nodes from your cluster.

4. On the machines that you’ve removed from the Service Fabric cluster, follow these steps:
   
a. Delete the contents of your Service Fabric data root (C:\ProgramData\SF) and your Service Fabric log root (C:\ProgramData\SF\Log).

b. Copy or download the standalone package for Service Fabric for Windows Server to the virtual machine (VM) or machine.
c. Unzip the package (C:\Temp\ServiceFabricStandalone).

d. Open Windows PowerShell as an admin.

e. Run the following commands.

```
IMPORTANT
Replace parameters as appropriate for your machine configuration.
```

```
cd C:\Temp\ServiceFabricStandalone
.
\AddNode.ps1 -NodeName AOS_12 -NodeType BatchOnlyAOSNodeType -NodeIpAddressOrFQDN 192.168.5.12 -ExistingClientConnectionEndpoint 192.168.5.21:19000 -UpgradeDomain ud0 -FaultDomain fd:/fd0 -X509Credential -ServerCertThumbprint 1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A -AcceptEULA -StoreLocation LocalMachine -StoreName MY -FindValueThumbprint 1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A1A
```

```
NOTE
When the node first reappears in Service Fabric Explorer, it might show the old node type. However, it should show the updated node type after a few minutes.
```


```
```

6. In the `fabricSettings` section, remove the `NodesToBeRemoved` parameter. When you've finished, the `fabricSettings` section should resemble the following example.

```
"fabricSettings": [
    {
      "name": "Setup",
      "parameters": [
        {
          "name": "FabricDataRoot",
          "value": "C:\ProgramData\SF"
        },
        {
          "name": "FabricLogRoot",
          "value": "C:\ProgramData\SF\Log"
        }
      ]
    }
]
```

7. Remove the following lines from the `Security` section.

```
"WindowsIdentities": {
  "\$id": "3"
},
```
8. Increment the version number of the configuration file. Make this change at the lowest increment. In the following example, the version number went from 1.0.1 to 1.0.2.

```
"ClusterConfigurationVersion": "1.0.2"
```

9. Save the configuration file.

10. Open Windows PowerShell as an admin, and run the following commands.

```
Connect-ServiceFabricCluster
Start-ServiceFabricClusterConfigurationUpgrade -ClusterConfigPath C:\Temp\ClusterConfig.json
Update-ServiceFabricClusterUpgrade -UpgradeReplicaSetCheckTimeoutSec 30
```

11. To monitor the upgrade, you can run the following command.

```
Get-ServiceFabricClusterUpgrade
```

12. Once the upgrade is finished from LCS use the Update Settings button without changing any settings to trigger an environment refresh. Alternatively you could also apply the latest hotfix.

```
IMPORTANT
This step causes downtime so be sure that your environment can be unavailable for some time.
```
Configure high availability for SQL Server Reporting Services (SSRS) nodes

This topic explains how to configure multiple Microsoft SQL Server Reporting Services (SSRS) nodes for Dynamics 365 Finance + Operations (on-premises) deployments.

High availability with Windows failover clusters

This scenario uses Windows failover clusters. Therefore, you will have one active node that receives all requests and one passive node that is idle. If the active node becomes unavailable, the cluster will detect this event, and the passive node will start to receive all network traffic.

This topic doesn’t cover the setup of Windows failover clusters. For information, see Create a failover cluster.

After the cluster is set up, you can configure your installation. The examples below will be based on the information displayed in the following illustration.

1. Update your configuration file (ConfigTemplate.xml):

   a. Update **ADServiceAccount** for the reporting service bootstrapper service.

   ```xml
   <ADServiceAccount type="gMSA" name="svc-ReportSvc$" refName="gmsaSSRS">
   <DNSHostName>svc-ReportSvc.contosoen05.com</DNSHostName>
   </ADServiceAccount>
   ```

   b. In the **ServiceFabricCluster** section, under **ReportServerType**, make sure that all your servers are listed.
c. Update the SSRS HTTPS certificate settings.

This example has been configured using the screenshot shown above. The Subject attribute should be set to the client access name. Additionally, for convenience, we have set the name and file name to be the same value. For the DNSName we have an entry for each of the preferred owners, as well as the client access name.

Even if you won’t be using the infrastructure scripts that are provided to generate the certificate, you should fill in the certificate information, because other scripts will rely on that information.

2. Follow the setup guide to complete the setup in your usual way.

If you’ve already created the Azure Service Fabric cluster, make sure that the nodes are added to it.

Rerun the Export-PfxFiles.ps1 script, and rerun the Complete-Prereqs.ps1 script on the appropriate machines, to ensure that the certificate for the SSRS web server is distributed to all the ReportServer nodes.

High availability with load balancers

In this scenario, a load balancer is configured to distribute requests among the different nodes that are available. These requests include all report generation requests.

When you set up this configuration, note that you must set up session affinity. The solution that you select must support this requirement. The type of session affinity that is required depends on the client. When the Application Object Server (AOS) node makes a request, the load balancer should direct all requests for that AOS node to the same SSRS node.
This topic doesn't include instructions for setting up a specific software load balancer or hardware load balancer.

Here is a general overview of this scenario:

1. Choose a load balancing strategy or product.
2. Configure the strategy or product according to your network topology.
3. Make sure that you've set up client (source IP) affinity.
4. Update the ConfigTemplate.xml file. Use the previous example as a guide.
5. Continue to set up the cluster in your usual way.

### IMPORTANT
If you've already created the Service Fabric cluster, make sure that the additional nodes are added to it.

Rerun the Export-PfxFiles.ps1 script, and rerun the Complete-Prereqs.ps1 script on the appropriate machines, to ensure that the certificate for the SSRS web server is distributed to all the ReportServer nodes.

### Deployed environments where the base deployment is earlier than Platform update 41

### NOTE
This configuration is supported only for Platform update 41 and later deployments.

If you want to enable high availability for the SSRS nodes in existing environments, you can use a predeployment script. For more information about predeployment scripts, see [Local agent pre-deployment and post-deployment scripts](#).

#### Predeployment script

**Invoke command example**

```powershell
Configure-SSRSAH.ps1 -AgentShare "\servername\D365FFOAgent" -Listener "LBDEN05FS1BI" -MachinesList "LBDEN05FS1BI1,LBDEN05FS1BI2" -TLSCertificateThumbprint "<cert thumbprint>" -ServiceAccount "contosoen05\svc-ReportSvc$"
```

### NOTE
These example values have been filled out according to the values used in the ConfigTemplate.xml file from the High availability with Windows failover clusters section.

#### Configure-SSRSAH.ps1 script

```powershell
param (
    [Parameter(Mandatory=$true)]
    [string]
    $AgentShare,

    [Parameter(Mandatory=$true)]
    [string]
    $Listener,

    [Parameter(Mandatory=$true)]
    [string]
    $MachinesList,

    [Parameter(Mandatory=$true)]
)
```
```powershell
[Parameter(Mandatory=$true)]
[string]
$TLSCertificateThumbprint,

[Parameter(Mandatory=$true)]
[string]
$ServiceAccount,

[string]
$ssrsServicePort = ""

$ErrorActionPreference = "Stop"

$basePath = Get-ChildItem $AgentShare\wp\"StandaloneSetup-*\ | Select-Object -First 1 -Expand FullName

if(!(Test-Path $basePath))
{
    Write-Error "Basepath: $basePath , not found" -Exception InvalidOperation
}

$configJsonPath = "$basePath\config.json"
$configJson = Get-Content $configJsonPath | ConvertFrom-Json
$updatedComponents = @()

foreach ($component in $configJson.components)
{
    if($component.name -eq "AOS")
    {
        $component.parameters.biReporting.persistentVirtualMachineIPAddressSSRS.value = $Listener
        $component.parameters.biReporting.reportingServers.value = $MachinesList
        $component.parameters.biReporting.ssrsUseHttps.value = "True"
        $component.parameters.biReporting.ssrsHttpsPort.value = $ssrsServicePort
    }
    elseif($component.name -eq "ReportingServices")
    {
        $component.parameters.enableSecurity.value = "True"
        $component.parameters.ssrsSslCertificateThumbprint.value = $TLSCertificateThumbprint
        $component.parameters.ssrsServerFqdn.value = $Listener
        $component.parameters.principalUserAccountType.value = "ManagedServiceAccount"
        $component.parameters.principalUserAccountName.value = $ServiceAccount
        $component.parameters.reportingServers.value = $MachinesList
        $component.parameters.ssrsHttpsPort.value = $ssrsServicePort
    }

    $updatedComponents += $component
}

$configJson.components = $updatedComponents
$configJson | ConvertTo-Json -Depth 100 | Out-File $configJsonPath

Write-Host "Successfully updated the configuration for SSRS HA."
```
You may want to secure the Dynamics 365 Finance + Operations (on-premises) environment behind a proxy. Proxy is a server that hides the actual servers that are serving traffic from the clients. The proxy server accepts requests from the clients on behalf of the environment and forwards the traffic to it. The clients are not aware of the actual servers that compose the environment. This adds another measure of security and enables load balancing.

Configure the proxy

Perform the following steps in each node of type OrchestratorType in the Microsoft Azure Service Fabric cluster.

1. Use remote access to connect to the Orchestrator virtual machine (VM).

2. Execute the following PowerShell script to retrieve the path of the machine.config file.

   ```
   <system.net>
   <defaultProxy enabled="true">
   <proxy <<<SET YOUR PROXY SETTINGS>> />
   </defaultProxy>
   </system.net>
   ```

3. Edit the machine.config file to add the following code example.

   ```
   <system.net>
   <defaultProxy enabled="true">
   <proxy <<<SET YOUR PROXY SETTINGS>> />
   </defaultProxy>
   </system.net>
   ```

4. Save the file.

5. Restart the virtual machine.

The above procedure must be performed for all Orchestrator node VMs.

Safe list URLs

The LocalAgent needs to communicate with Azure resources. As a result, the following URLs should be added to a safe list on the proxy or firewalls so that all OrchestratorType nodes can access them:

- lcsapi.lcs.dynamics.com
- login.windows.net
- uswelcs1lcm.queue.core.windows.net
- www.office.com
- login.microsoftonline.com
- dc.services.visualstudio.com
- uswelcs1lcm.blob.core.windows.net
The Microsoft Dynamics 365 team monitors the health and performance of the Azure Services that provide functionality for our cloud-based customers by using state-of-the-art Azure diagnostic tools. For customers who have implemented Finance + Operations (on-premises) and would like to have the ability to monitor the health and performance of their on-premises solution, there are several third-party offerings available.

This topic describes the setup and configuration of Elastic Stack, a third-party product, and one of many choices that can provide diagnostic monitoring of your on-premises solution.

When you consider a diagnostic solution, consider the following fundamentals of your implementation:

- Your diagnostic system should be able to collect and store 30 days’ worth of diagnostic information.
- Your diagnostic repository should be set up in a central location that is sharable among many client computers.
- Create structured diagnostics events, including event type, classification, and data.
- Events stored in raw text (deserialized) can be easily queried and searched.
- Avoid storing sensitive or personal data in events.

By default, communication in an Elastic Stack cluster is not sent over HTTPS. Don't set up the Elastic Stack unless you've considered the risks, and prepared or implemented mitigations for those risks. The paid version of X-Pack can be used to encrypt communication in the Elastic Stack. For setup information, see Setting up TLS on a cluster. There is also an open source Elasticsearch plug-in. Although Microsoft hasn't tested this plug-in, according to the documentation, it can enable HTTPS. Microsoft recommends that you always utilize encrypted communication using HTTPs, VPN, or another secure, encrypted protocol. Many industry certifications and laws require the use of encrypted transmission if your content includes end user, customer, personal, or sensitive data.

**Diagnostic data guidelines**

To diagnose the deployment and execution of Finance + Operations (on-premises), you must have access to diagnostic data. For a cloud deployment, Microsoft stores and monitors the diagnostic data from services to help keep the environment healthy. For an on-premises deployment, the customer is responsible for this task.

You can select the diagnostic data store and query tool that you prefer to use. However, at a minimum, the tool should perform the following tasks:

- The store should be able to store 30 days' worth of diagnostic data.
- The events should be stored in a centralized location, so that support engineers don't have to switch between multiple machines to find events that are relevant to an issue.
- The events should be discoverable based on event type and event data.
- The event data (in XML format) should be deserialized so that the event data can be queried on and traversed.

**Elastic Stack example**

To meet the diagnostic data guidelines that are listed in the previous section, Microsoft tested the Elastic Stack setup. This setup includes the following components:
- **Elasticsearch** – For storage, event indexing, and event querying. For more information about Elasticsearch, see the [Elastic website](https://www.elastic.co).
- **Logstash** – For load distribution and event data transformation.
- **Winlogbeat** – For diagnostic data collection.
- **Kibana** – An interface for querying the data that is stored in Elasticsearch.

**NOTE**

By default, communication in an Elastic Stack cluster is **not** sent over HTTPS. Don't set up the Elastic Stack unless you've considered the risks, and prepared or implemented mitigations for those risks. The [paid version](https://www.elastic.co) of X-Pack can be used to encrypt communication in the Elastic Stack. For setup information, see [Setting up TLS on a cluster](https://www.elastic.co). There is also an open source [Elasticsearch plug-in](https://www.elastic.co). Although Microsoft hasn't tested this plug-in, according to the documentation, it can enable HTTPS.

If you deploy the Elastic Stack, your experience might vary if you follow the steps that are described in this topic. For its tests, Microsoft used version 6.2.3 of the Elastic Stack components and Microsoft Dynamics 365 for Finance and Operations 7.3 with platform update 12.

This topic describes how Microsoft handled the setup and configuration steps that are required for the Elastic Stack to work for an on-premises deployment. For guidance that isn't related to Finance + Operations (on-premises), see the documentation on Elastic.co.

### Install and configure the Elastic Stack

All hosted components of the Elastic Stack, except Winlogbeat, run on Java. For the test scenario, Microsoft first downloaded and installed the latest version of Java Runtime Environment (JRE) 8 (64-bit) on each node that will run Elasticsearch, Logstash, or Kibana (that is, all the Orchestrator nodes). You can get Java 8 from [https://www.oracle.com/technetwork/java/javase/downloads/index.html](https://www.oracle.com/technetwork/java/javase/downloads/index.html).

As of June 2018, the Elastic Stack runs on Java 8. Any attempt to run it on a newer version of Java might not work.

**NOTE**

The whole Elastic Stack, except Winlogbeat, can be hosted on Linux. For its tests, Microsoft hosted the stack on Microsoft Windows Server 2016 virtual machines (VMs).

Remember to open ports in the firewall for the various components on each node.

If you get stuck during setup, Elastic.co has extensive and well-written documentation about the installation and configuration of the Elastic Stack. For help with specific types of errors, web searches yield reliable results from both the Elastic.co forum and StackOverflow.

### Component matrix

For its tests, Microsoft used the following setup for a small to medium-sized deployment.

<table>
<thead>
<tr>
<th>NODE</th>
<th>ELASTICSEARCH</th>
<th>LOGSTASH</th>
<th>KIBANA</th>
<th>WINLOGBEAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orchestrator #1</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Orchestrator #2</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Orchestrator #3</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
For testing purposes, Microsoft used the Orchestrator machines for the ELK installation. Because it can take up critical resources from the Orchestration services, don't use the Orchestrator machines for ELK installations on production environments or critical Sandbox Environments. Instead, use separate machines to host the ELK services.

**Elasticsearch**

The installation of Elasticsearch is fairly straightforward. For its tests, Microsoft downloaded the Microsoft Windows Installer (MSI) file onto the Orchestrator #1 and Orchestrator #2 nodes. Most of the default settings in the installer can be left as is. This section describes the settings that Microsoft changed.

To facilitate Elasticsearch to start running again if the operating system (OS) is restarted, Microsoft installed it as a service on Windows. The installer can be used to set up the service.

On the **Configuration** page of the installer, Microsoft used the same cluster name when it installed each Elasticsearch node in the cluster.

Microsoft set every Elasticsearch node to perform all three roles: Data, Master, and Ingest.

Depending on the amount that you expect Kibana and Elasticsearch to be used, consider increasing the memory usage. You can change this setting later by modifying the -Xm options in the C:\ProgramData\Elastic\Elasticsearch\config\jvm.options file and restarting Elasticsearch.

Depending on the number of Elasticsearch nodes that you set up, you can set the Discovery minimum master nodes appropriately. If you aren't sure, you can keep the master nodes empty. For more information about discovery and nodes, see **Node**.

For discoverability, in **Network settings**, Microsoft set the **Network host** value of each node to that node's IP address and added the IP addresses of all Elasticsearch nodes to the **Unicast Hosts** list for each node. For example, for Orchestrator #1, which has the IP address 10.0.0.12, Microsoft set the **Network host** value to 10.0.0.12 and added the following IP addresses to the **Unicast Hosts** list: 10.0.0.12 and 10.0.0.13, where 10.0.0.13 is Orchestrator #2.

*If you’re installing Elasticsearch version 6.3 or higher, you can disregard this paragraph.* You can install X-Pack either now or later. For more information about setup and whether you should install X-Pack, see the "X-Pack" section of this topic. For now, unless you know what X-Pack is for, don't install it.

**IMPORTANT**

Open the HTTP port (by default, port 9200) and the node communication port (by default, port 9300) in your firewall.

To verify that the installation was successful, start a browser, and open the application address. You should see some JavaScript Object Notation (JSON) output.

**Logstash**

In its test setup, Microsoft found that some events from Winlogbeat required adjustments. Logstash provides that functionality.

Microsoft downloaded Logstash to C:\ELK\Logstash on the Orchestrator #2 and Orchestrator #3 nodes.

To help ensure that Logstash runs on startup, we used the Non-Sucking Service Manager (NSSM) to set up a
service for the Logstash batch script.

1. Copy nssm.exe to the Logstash bin folder (for example, C:\ELK\Logstash\6.2.4\bin\).

2. Open Windows PowerShell from the bin folder, and run the following command.

```
.
ssm.exe install Logstash
```

3. On the Application tab, set the following fields, and then click Save:

   - **Path:** C:\ELK\Logstash\6.2.4\bin\logstash.bat
   - **Startup directory:** C:\ELK\Logstash\6.2.4
   - **Arguments:** -f C:\ELK\Logstash\config\logstash-dyn365finops.conf

   (There are more settings that you can set. However, these settings suffice for now.)

4. Run the following command.

```
.
ssm.exe start Logstash
```

In the tests that Microsoft performed, NSSM had trouble restarting the installed services. Because NSSM wasn't 100-percent reliable for Logstash and Kibana, the service was treated as an OS startup service and little else.

Microsoft created a configuration file for Logstash called logstash-dyn365finops.conf. This file is available in the LCS Shared Asset library, under the Model asset type in a zipped file called "LBD Diagnostic configurations". Go to the LCS Shared Asset library to download this file. After you extract it, you need to put it in C:\ELK\Logstash\6.2.4\config. This file performs useful transformations on diagnostics.

To make the configuration work for your setup, you must change the **hosts** fields in the **output** section so that they point to the Elasticsearch nodes in your cluster. For example, change **hosts** to ["ORCH1:9200", "ORCH2:9200"].

The configuration was tested by using the Winlogbeat configuration from the next section.

Remember to open the Winlogbeat port (by default, port 5044) in your firewall on the machine that is hosting Logstash, so that Beats can send data to Logstash.

**Winlogbeat**

Microsoft downloaded Winlogbeat to each Application Object Server (AOS) and Orchestrator node at C:\ELK\Winlogbeat, and configured the winlogbeat.yml file. The sample configuration file for Winlogbeat is available in the LCS Shared Asset library, under the Model asset type in a zipped file called "LBD Diagnostic configurations". Go to the LCS Shared Asset library to download this file.

To make the configuration work for your set up, you must change the **output.logstash.hosts** fields so that they point to all your Logstash nodes. Winlogbeat handles the load balancing.

When Winlogbeat runs on an Orchestrator node, the **Tags** field can be changed from **AOS** to **ORCH** or a similar value. Microsoft also used the **fields.env** field to set the environment of the deployment (sandbox, sandbox-n, or production). In this way, there is a cleaner separation when data from multiple environments and node types is queried.

Winlogbeat includes a service installer. Microsoft used this installer to set up Winlogbeat as a service on each node. Press the Windows logo key+R to start the Run tool, and then run the following command.

```
powershell.exe -ExecutionPolicy Bypass -File C:\ELK\Winlogbeat\install-service-winlogbeat.ps1
```
Kibana

Kibana provides the interface to query the diagnostic data in Elasticsearch.

Microsoft downloaded Kibana to C:ELK\Kibana and configured the kibana.yml file in the following manner.

```
server.host: "10.0.0.14"
server.name: "Dyn365FinOps On-Premises Diagnostics"
elasticsearch.url: "http://ORCH1:9200"
```

From Kibana, Microsoft had to define index patterns on the **Management** tab. Because index patterns group indexes by name, an index pattern was required for the two indexes that were made: deployment-* and runtime-*. The index names are case-sensitive.

Microsoft set the runtime-* index pattern as the default pattern. When you're looking at the index patterns on the **Management** tab, click the asterisk (*). The index pattern will then appear on the **Discover** tab.

Microsoft ran Kibana as a service in the same manner as Logstash, so that Kibana is started at OS startup. Unlike Logstash, kibana.bat doesn't need the path of the configuration files. Therefore, you can just install an NSSM service that points to C:ELK\Kibana\6.2.4\bin\kibana.bat.

If you want users to browse Kibana on your network, remember to open the port for Kibana. The default port is 5601.

### Example queries on the Discover tab in Kibana

The following sample queries can help you start probing the diagnostic data. If you require something more than the examples show, you can try one of the following queries:

- **Find slow database queries**: Enter slow in the search field to find events that have the word “slow” somewhere in the event data. If you want to be more precise, you can find events that have a task name of AosDatabaseSlowQuery and then enter TaskName:AosDatabaseSlowQuery in the search field.

- **Find recent exceptions**: Enter exception in the search field to find events that have either thrown an exception, or handled an exception and logged it. In the upper-right corner of Kibana, you can select the time frame that the search should be limited to. The time frame that you set there is persisted between tabs. Therefore, the data on the **Visualize** tab will reflect the selected time frame.

- **Find events from an AOS node**: Enter host:AOS1 in the search field to find all events from that node.

- **Find events with proximity, in time, to another**: When you've found an event that you're interested in, click View surrounding documents next to the header of that event to find events that occurred at the same time. If you see events that occurred at around the same time but from different AOS nodes, you can add additional filtering to view only events from the node that you want.
Thirty-day data retention
To keep its hard disks free from stale data, Microsoft used Curator v5.5 to clean up indexes that were older than 30 days.

Microsoft downloaded Curator to one of the Orchestrator nodes at C:\ELK\Curator. The sample configuration file, curator.yml, available in the LCS Shared Asset library, under the Model asset type in zipped file "LBD Diagnostic configurations", was then put in C:\ELK\Curator to connect Curator to its Elasticsearch cluster. You'll need to edit the file to reference your specific servers.

Curator runs actions, and Microsoft created an action configuration file called "30day_data_retention_actions.yml" to clean up 30-day-old indexes in C:\ELK\Curator. The retention configuration file is available in the LCS Shared Asset Library, under the Model asset type in a zipped file called "LBD Diagnostic configurations". Go to the LCS Shared Asset Library to download this file.

Microsoft created a basic task in Windows Task Scheduler. This task has a weekly trigger on Saturday and Sunday, and the trigger has the following settings to start a program:

- **Program/script:** C:\ELK\Curator\curator.exe
- **Add arguments:** --config curator.yml \30day_data_retention_actions.yml
- **Start in:** C:\ELK\Curator

X-Pack

**IMPORTANT**
As of June 2018, Elastic Stack components have been released that start with version 6.3. This updated version handles X-Pack in a more graceful manner, by enabling the free features of X-Pack by default, without requiring that you update the license every year, and by letting you opt in to the paid features afterward. If you install an Elastic Stack version that is earlier than 6.3, the content in this section only partially applies to the setup.

You can select to install X-Pack when you install Elasticsearch. Alternatively, you can install it later.

X-Pack has a free basic license that must be updated every year.

Microsoft installed the free version to enable query data to be exported from Kibana in comma-separated value (CSV) format. There are other useful features in X-Pack, but some are available only in a paid subscription.

To enable only the free features and avoid using other X-Pack trial features, Microsoft added the following settings to our elasticsearch.yml and kibana.yml configuration files:

```yaml
xpack.graph.enabled: false
xpack.ml.enabled: false
xpack.security.enabled: false
xpack.watcher.enabled: false
```

These settings also stop Kibana and Elasticsearch from asking for credentials because the security module is no
For X-Pack to work, the logstash.yml configuration file must also be configured in the following manner.

```
xpack.monitoring.elasticsearch.url: "http://orch1:9200"
```

The paid version of X-Pack includes HTTPS encryption for connections throughout the cluster, password-protected data access, and more. For more information about X-Pack, see the Elastic website.

**Export a query to a CSV file**

In Kibana, on the Discover tab, write a query, and save it. After you save the query, on the Reporting tab at the top of the Discover page, click Generate CSV.

**Troubleshooting**

**You don’t receive any data in Kibana**

If you don’t receive any data in Kibana, review the logs from Winlogbeat to Logstash, Elasticsearch, and Kibana. Note that the Winlogbeat installation puts its logs in C:\ProgramData\winlogbeat\Logs, whereas the other Elastic Stack components put their logs close to the installation path (for example, in C:\ELK\Elasticsearch\logs).
Disaster recovery is an important consideration for on-premises deployments of Dynamics 365 Finance + Operations (on-premises) to protect from events that could put your organization’s operations at risk. Examples of such events include equipment failures, datacenter break downs due to cyberattacks, electrical, physical, or other disasters.

The core concept of disaster recovery involves the use of a second datacenter including a data recover environment. We recommend that you plan, document, and test disaster recovery as carefully as your production setup.

Limitations of this content
This topic does not cover specific configuration details for disaster recovery of the following components:

- Active Directory Federation Services (AD FS)
- File storage
- SQL Server

NOTE
High availability configuration isn't covered in this topic. For more information about the minimum setup required for high availability, see System requirements for on-premises deployments.

Recommendations
Remember to keep your disaster recovery environment updated with the latest Windows Updates. Your environment should have the latest security updates and not require updates during a disaster event.

Ensure that you're applying new prerequisites that are specified by Microsoft. Also, keep your Service Fabric cluster updated and perform certificate rotations as required.

After you've read through this topic, document the steps that need to be taken by your team. After you've done that, go through the steps multiple times to ensure that you don't encounter unexpected problems and you minimize the potential downtime.

Overview
The basic configuration for disaster recovery involves deploying a duplicate of the production environment within another datacenter (the secondary datacenter) and replicating databases to that datacenter. If a disaster event takes place, a few manual steps can be taken to bring the environment that is within the secondary datacenter online.

The following diagram illustrates the required setup, at a high level.
Environment configuration

In Lifecycle Services (LCS), the production environment should be deployed using the environment slot named **Production**. Your disaster recovery environment will not use an additional environment slot in LCS. It will instead reuse the slot for your production environment.

Finance and Operations AOS nodes and SQL Server must be co-located within the same datacenter. For more information, see [System requirements for on-premises deployments](#).

Deploying code packages to production

When code packages are deployed to the production environment, they don't need to be deployed to the disaster recovery environment. That environment should be unused and no Service Fabric services should be deployed.

Environment deployment settings

The disaster recovery environment should have a similar configuration as the production environment. The following table illustrates the shared and specific settings for disaster recovery.

<table>
<thead>
<tr>
<th>ENVIRONMENT SETTINGS</th>
<th>DISASTER RECOVERY ENVIRONMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Directory settings</td>
<td></td>
</tr>
<tr>
<td>Administrator user</td>
<td>Same as production</td>
</tr>
<tr>
<td>AD FS URL</td>
<td>Same as production</td>
</tr>
<tr>
<td>AD FS OpenId Connect client ID for AOS</td>
<td>Same as production</td>
</tr>
<tr>
<td>AD FS OpenId Connect client ID for Financial Reporting</td>
<td>Same as production</td>
</tr>
<tr>
<td>SQL Database configuration</td>
<td></td>
</tr>
<tr>
<td>ENVIRONMENT SETTINGS</td>
<td>DISASTER RECOVERY ENVIRONMENT</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>SQL Server name</td>
<td>Same as production</td>
</tr>
<tr>
<td>AX database name</td>
<td>Same as production</td>
</tr>
<tr>
<td>Financial Reporting database name</td>
<td>Same as production</td>
</tr>
<tr>
<td><strong>File share settings</strong></td>
<td></td>
</tr>
<tr>
<td>File share for document store</td>
<td>Same as production</td>
</tr>
<tr>
<td>File share certificate thumbprint</td>
<td>Same as production</td>
</tr>
<tr>
<td><strong>SSRS configuration settings</strong></td>
<td></td>
</tr>
<tr>
<td>IP address of SSRS instance</td>
<td>Can be different</td>
</tr>
<tr>
<td>SSRS certificate thumbprint</td>
<td>Same as production</td>
</tr>
<tr>
<td><strong>Configure service settings</strong></td>
<td></td>
</tr>
<tr>
<td>DNS host name of Dynamics 365 instance</td>
<td>Can be different</td>
</tr>
<tr>
<td>AOS service user</td>
<td>Same as production</td>
</tr>
<tr>
<td>MR application service user</td>
<td>Same as production</td>
</tr>
<tr>
<td>MR process service user</td>
<td>Same as production</td>
</tr>
<tr>
<td>MR click-once service user</td>
<td>Same as production</td>
</tr>
<tr>
<td><strong>Application certificate settings</strong></td>
<td></td>
</tr>
<tr>
<td>Data encryption certificate thumbprint</td>
<td>Same as production</td>
</tr>
<tr>
<td>Data signing certificate thumbprint</td>
<td>Same as production</td>
</tr>
<tr>
<td>Session authentication certificate thumbprint</td>
<td>Same as production</td>
</tr>
<tr>
<td>SSL certificate thumbprint</td>
<td>Same as production</td>
</tr>
<tr>
<td>Management reporter certificate thumbprint</td>
<td>Same as production</td>
</tr>
</tbody>
</table>

1 SSRS is referenced by IP. If the exact machine IP can’t be configured in the disaster recovery environment, the IP can be different.

2 This depends on your network configuration. If you have a load balancer that can handle diverting traffic to the other environment, then the host name can be the same. If you’re unable to do that, then use a different host name.

**SQL Server Always-On Availability configuration**
The business data database (AXDB) should be replicated to the secondary datacenter, typically using SQL Server Always-On availability groups feature. For more information, see Always On availability groups.

<table>
<thead>
<tr>
<th>DATABASE</th>
<th>REPLICA TED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business data (AXDB)</td>
<td>Yes</td>
</tr>
<tr>
<td>Financial Reporting</td>
<td>Yes</td>
</tr>
<tr>
<td>BYODB</td>
<td>Yes</td>
</tr>
<tr>
<td>OrchestratorData</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Failing over to disaster recovery

**Overview**

When a disaster event occurs, the primary datacenter may be unavailable but within the secondary datacenter, the following components will be available.

At the initial moment of the disaster event, the disaster recovery environment will be empty. The only thing present will be a configured Service Fabric cluster and SQL Server, which contain all of the replicated production data.
To bring the disaster recovery environment online, you'll need to have LCS deploy what is currently available in your Production environment into the disaster recovery environment.

**IMPORTANT**
Before you continue, ensure that no Dynamics Service Fabric services are running in your production environment (in case you're only failing due to a partial disaster event).

**Deploy the LocalAgent**
Download the LocalAgent installer and configuration file from LCS to your disaster recovery environment. After you have the configuration file, open it. Ensure that the connectionEndpoint under the serviceFabric section points to the IP or FQDN of a server in the disaster recovery environment. After modifying the file, save it locally and deploy the LocalAgent as you typically would.

**IMPORTANT**
Do not make changes to your connector settings in LCS.

Until your main production environment comes back online, this LocalAgent will process all requests that LCS puts into the message queue. That's why it's important that you ensure no services are running in your production environment. Eventually, when your orchestrator nodes come back up in your primary datacenter, unprovision the LocalAgent from the cluster.

**Caution**
The LocalAgent must only be running in one datacenter at a time. At this point it should only be running in your secondary datacenter.

**Prepare your pre-deployment scripts (optional)**
Pre-deployment scripts are necessary when changes to the deployment configuration are required. This script will have to modify the config.json file with the values you specify. It will be the customers' responsibility to create this script.

You can find the location of the config.json file by running the following command.

```sql
select Location from DeploymentInstanceArtifact where AssetId='config.json' and DeploymentInstanceId = 'LCSENVIRONMENTID'
```

**NOTE**
Replace LCSENVIRONMENTID with the ID of your environment. You can obtain this ID from the full details page for your environment in LCS.

If the SSRS node IP is different, you'll have to modify the following values.

```json
"biReporting": {
  "persistentVirtualMachineIPAddressSSRS": {
    "value": "192.168.5.31"
  },
  "reportingServers": {
    "value": "192.168.5.31"
  }
},
```

If you are changing the host name, the following modifications are required.
"name": "AOS",
"parameters": {
    "activeDirectory": {
        "aadValidAudience": {
            "value": "https://ax.contosoen05.com/",
        },
        "infrastructure": {
            "hostName": {
                "value": "ax.contosoen05.com"
            },
        }
    },
    "name": "FinancialReporting",
    "parameters": {
        "aad": {
            "cookieDomain": {
                "value": "ax.contosoen05.com"
            },
            "validAudiences": {
                "value": "https://ax.contosoen05.com/",
            }
        }
    }
}

**IMPORTANT**
If you are changing the hostname URL for your deployment, ensure that your AD FS server is configured to accept the new URL. For more information, see [Reuse the same AD FS instance for multiple environments](#).

If the file share is shared between the production and disaster recovery environments, this pre-deployment script should be disabled. Only enable it when deploying to your disaster recovery environment.

**Ensure reports get deployed**
Because the database has previously been synchronized successfully, synchronization typically would be skipped. However, to synchronize the reports because the SSRS node is empty, perform the following actions.

**Version 10.0.13 or later**
Run the following command against your business data database (AXDB):

```
UPDATE SF.synclog SET STATE=5, SyncStepName = 'ReportSyncstarted' WHERE CODEPACKAGEVERSION in (SELECT TOP(1) CODEPACKAGEVERSION FROM SF.SYNCLOG ORDER BY CREATIONDATE DESC)
```

**Version 10.0.12 or earlier**
Run the following command against your business data database (AXDB):

```
DELETE FROM SF.synclog WHERE CODEPACKAGEVERSION in (SELECT TOP(1) CODEPACKAGEVERSION FROM SF.SYNCLOG ORDER BY CODEPACKAGEVERSION DESC)
```

**NOTE**
If you are using version 10.0.12 or earlier, a full database synchronization will be executed.

**Deploy your environment**
To deploy your environment, follow these instructions.

1. In LCS, go to the environment page for your production environment.

2. Select **Maintain** and then select **Update Settings**.

3. Don't change any values. Select **Prepare**.

4. After downloading is finished and preparation is completed, the **Update environment** button will be displayed. Select this button to start updating your environment.

5. After the environment is deployed, the disaster recovery environment is ready for use.

**Using your disaster recovery environment**

You can use your disaster recovery environment as you typically would, except that updates or hotfixes shouldn't be applied to the environment. If you must apply updates to your environment, your failback process will differ from the one described below. Failing back under this condition is not covered in this topic.

**Failing back to your production environment**

**IMPORTANT**

At this point, no Dynamics Service Fabric services should be running in your production environment.

Secure a downtime window in which you can switch operation from the disaster recovery environment to the Production environment. In the downtime window, disable all non-Orchestrator nodes in the disaster recovery environment through Service Fabric Explorer. Once all nodes are disabled, failover your SQL Server to the production datacenter.

After the failover has occurred, start the AOS, SSRS, and MR nodes in your primary datacenter. Carry out validation tests to ensure that your environment is functioning as expected. When you determine that your environment is working as expected, remove the LocalAgent from your disaster recovery environment and
reinstall it on your Production environment.

Clean up your DR environment by manually unprovisioning all Dynamics Service Fabric services.

Caution
Do not use the Cleanup functionality in LCS to perform the clean up of your disaster recovery environment.

Failback checklist
- Non-orchestrator nodes are disabled in disaster recovery datacenter.
- SQL Server is failed back to primary datacenter.
- LocalAgent is uninstalled in your disaster recovery datacenter.
- All Dynamics Service Fabric services (including LocalAgent) are running in your primary datacenter.
- No Dynamics Service Fabric services are deployed in your disaster recovery datacenter.

IMPORTANT
Your primary environment will be functioning as usual and can be serviced after you ensure that all items in the checklist are verified.
The cloud version provides several features that allow for deeper integration with Microsoft Power BI. Some of these features haven’t yet been implemented for on-premises deployments. However, the availability of Entity Store in on-premises deployments lets customers use PowerBI.com to report on and analyze data.

This topic outlines the analytical capabilities that are available in on-premises deployments that run Microsoft Dynamics 365 for Finance and Operations platform update 26 (May 2019) and later.

<table>
<thead>
<tr>
<th>FEATURE/CAPABILITY</th>
<th>CLOUD</th>
<th>ON-PREMISES (PLATFORM UPDATE 26 OR LATER)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analytical workspaces</td>
<td>Available</td>
<td>Not yet implemented</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Customers can use the reports that are built on Entity Store together with PowerBI.com.</td>
</tr>
<tr>
<td>Pin reports, tiles, and dashboards from PowerBI.com into the client.</td>
<td>Available</td>
<td>Not yet implemented</td>
</tr>
<tr>
<td>Author and distribute Power BI reports that use application data.</td>
<td>Available</td>
<td>Available</td>
</tr>
<tr>
<td>Extract application data into data warehouses.</td>
<td>Available</td>
<td>Available</td>
</tr>
</tbody>
</table>

**Enable Entity Store on-premises**

This topic supplements the Set up and deploy on-premises environments (Platform update 12 and later) topic. The section numbers that follow correspond to the section numbers in that topic.

**3. Plan your users and service accounts**

<table>
<thead>
<tr>
<th>USER ACCOUNT</th>
<th>TYPE</th>
<th>PURPOSE</th>
<th>USER NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>AOS SQL AXDW DB Admin user</td>
<td>SQL user</td>
<td>The application uses this user to enter information in the AXDW database. This user is optional and must be created only if you want Entity Store support.</td>
<td>axdwadmin</td>
</tr>
</tbody>
</table>
14. Configure the databases

The steps in this section are optional.

If you want to create a database that can be used for Entity Store, you must first modify the ConfigTemplate.xml file.

1. Under **DbServer – Security**, set the **generateUser** flags for **axdwadmin** and **axdwruntimeuser** to **True**. The scripts that you run in the next step will then create those two users. You will be prompted to set passwords for the users.

2. Run the following scripts.

   ```
   .\Initialize-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName EntityStore
   .\Configure-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName EntityStore
   ```

The Initialize-Database.ps1 script performs the following actions:

- Create an empty database that is named **AXDW**. This database is used for Entity Store.
- Create a new user that is named **axdwadmin** and that has SQL authentication enabled, and prompt you for the user’s password.
- Create a new user that is named **axdwruntimeuser** and that has SQL authentication enabled, and prompt you for the user’s password.
- Grant the axdwadmin and axdwruntimeuser users **db_owner** permissions on the database.

The Configure-Database.ps1 script performs the following actions:

- Set the specified database file and log settings.
- GRANT VIEW SERVER STATE TO axdwadmin.
- GRANT VIEW SERVER STATE TO axdwruntimeuser.

15. Encrypt credentials

Create a `Credentials.json` file as shown here. The **AosDWAuth** category is optional and is used only if Entity Store is enabled.
Here is an explanation of the preceding code lines:

- **AccountPassword** is the encrypted domain user password for the Application Object Server (AOS) domain user (contoso\axserviceuser).
- **SqlUser** is the encrypted SQL user (axdbadmin) that has access to the application database (AXDB). **SqlPwd** is the encrypted SQL password.
- **DWUser** is the encrypted SQL user (axdwadmin) that has access to the Entity Store database (AXDW). **DWPwd** is the encrypted SQL password that is entered when the Initialize-Database.ps1 script is run.
- **DWRuntimeUser** is the encrypted SQL user (axdwruntimeuser) that has access to the Entity Store database (AXDW). **DWRuntimePwd** is the encrypted SQL password that is entered when the Initialize-Database.ps1 script is run.

More information

Entity Store was enabled in Platform update 26.

Creation of the Entity Store database and users is optional. When you configure a deployment in Microsoft Dynamics Lifecycle Services (LCS), you can leave the Entity Store customizations blank.

If Entity Store should be enabled in an environment, the following tasks must be completed:

- Create an **AXDW** database by using the scripts that are described in step 14.
- Create **axdwadmin** and **axdwruntimeuser** users that have appropriate privileges, by using the scripts in that are described step 14. Users are defined in the following files: ConfigTemplate.xml and DatabaseTopologyDefinition.xml.
- Create a **Credentials.json** file as described in step 15. User names and passwords should be defined for axdwadmin and axdwruntimeuser as described in step 14, and they should be encrypted.
- The Entity Store SQL Server name and Entity Store database name must be filled in during deployment in LCS.
  - The database name is created in step 14 and is defined in the DatabaseTopologyDefinition.xml file. The default name is **AXDW**.
  - The SQL Server name is the fully qualified domain name (FQDN) of the Microsoft SQL Server or Always on listener. An example of this FQDN is **sqlinstance.onprem.contoso.com**. It's the server that the AXDW database is created on.

To enable Entity Store in an environment that has already been deployed, you can use the **Update Settings** action under the **Maintain** button in LCS. This will open a dialog that will allow you to specify the Entity Store configuration.
Authoring and distributing reports by using Entity Store on-premises

Entity Store is an operational data warehouse that is included. It lets power users and business analysts author reports that use simplified and enriched data. Entity Store contains aggregate measurements, which are simplified star schemas. For more information about how to author reports by using Entity Store, see Create analytical reports by using Power BI Desktop.

Note the following additional steps that are specific to on-premises deployments:

- For on-premises environments, you don’t have to use LCS to deploy Power BI reports to production or sandbox environments. Because admins can point PowerBI.com datasets to specific Entity Store databases in on-premises environments, you don’t have to use the functionality that LCS offers. However, you might have to configure the on-premises gateway to enable PowerBI.com to access data on-premises. For more information about the gateway, see Power BI gateway documentation.

- Although cloud-based application environments support only reports that are authored by using the DirectQuery option, on-premises Entity Store supports both DirectQuery reports and Import mode reports.

- Analytical workspaces aren’t yet implemented in on-premises deployments. Instead of viewing reports in analytical workspaces, you can deploy them to PowerBI.com environments. The reports can then be used by users who have access to PowerBI.com. Your users might require appropriate licenses to access reports on PowerBI.com.
This topic provides information about how to reconfigure your environment with a new platform or topology and how to update the configuration of your existing environment.

Prerequisites

Before you complete the steps in this topic, you must update your local agent. For more information, see the topic, Update the local agent. The procedure in this topic will only work with local agents that are on or above version 1.1.0.

Reconfigure your environment

1. In Lifecycle Services (LCS), navigate to your on-premises project and open the Environments blade.

2. Do one of the following based on whether you are going to reconfigure or take a new deployment.

   - If you need to reconfigure your environment, click Reconfigure.
   - If you need to take a new deployment or topology, select the new topology for your platform and enter the environment name. You can use the same name or enter a new one.

3. Click Advanced Settings to update your configuration. The configuration from your previous deployment will be saved.
At some point, you might have to redeploy your on-premises environment. This could be to apply a new platform update or because of changes or issues in your implementation. Before you delete the environment you are currently working with, you should save your configuration setting information to use when you redeploy. This topic describes how to save configuration settings and how to redeploy your environment.

Save your configuration

Before you delete the environment you plan to update, use the following steps to save your configuration.

1. In LCS, navigate to **Project Settings > On-prem Connectors**.
2. Select the connector to your environment, and then click **Edit**.
3. On the **Edit connector** tab, navigate to **Configure Agent > Enter Configuration**.
4. Copy the value of the Download Fileshare location in the **Configuration Settings** section. You will need this later.
5. Log in to the on-premises environment file share machine and copy the `\agent\wp<environment name>\StandaloneSetup\config.json`. You can use the configuration settings in this json file to redeploy your environment.

**Configuration settings**

The following tables provide information about configuration settings. Use the **Configuration setting** value from the .json file that you saved in the previous procedure.

### Active Directory Federation Services settings

<table>
<thead>
<tr>
<th>FIELD</th>
<th>CONFIGURATION SETTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>The email address of the user who will be the initial administrator (such as, <a href="mailto:adminuser@yourdomain.com">adminuser@yourdomain.com</a>)</td>
<td>components. (AOS).parameters.provisioning.adminPrincipalName.value</td>
</tr>
<tr>
<td>ADFS OpenID metadata endpoint for the Dynamics 365 Application group. (such as, https://[federation-service-name]/adfs/.well-known/openid-configuration)</td>
<td>components. (AOS).parameters.activeDirectory.adfsMetadata.value</td>
</tr>
<tr>
<td>ADFS OpenID Connect client ID for the AOS application group</td>
<td>components. (AOS).parameters.activeDirectory.adfsClientId.value</td>
</tr>
<tr>
<td>ADFS OpenID Connect client ID for the Financial Reporting application group</td>
<td>components. (FinancialReporting).parameters.aad.nativeClientAuthentication.clientId.value</td>
</tr>
</tbody>
</table>

### SQL database configuration

<table>
<thead>
<tr>
<th>FIELD</th>
<th>CONFIGURATION SETTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL SERVER</td>
<td>components. (AOS).parameters.database.dbServer.value</td>
</tr>
<tr>
<td>AX DATABASE</td>
<td>components. (AOS).parameters.database.dbName.value</td>
</tr>
</tbody>
</table>
### Financial Reporting Database Components

**FIELD**

- FINANCIAL REPORTING DATABASE

**CONFIGURATION SETTING**

- components. (FinancialReporting).parameters.mrdb.dbName.value

### File Share Settings

**FIELD**

- The file share path for the Microsoft Dynamics 365 instance.
- The file share certificate thumbprint for the Microsoft Dynamics 365 instance.

**CONFIGURATION SETTING**

- components.(AOS).parameters.storage.fileSharePath.value
- components.(AOS).parameters.storage.sharedAccessThumbprint.value

#### NOTE

When you copy the file path configuration value from .json file to LCS UI, make sure to remove the extra backslashes. For example, configuration value `\\DC1\D365FFOStorage` from the .json file should be `\DC1\D365FFOStorage` in the LCS UI.

### SSRS Configuration Settings

**FIELD**

- The IP Address of the SSRS instance.
- The thumbprint used by the SSRS application to communicate with AX Service.

**CONFIGURATION SETTING**

- components. (AOS).parameters.biReporting.persistentVirtualMachineIPAddress.value
- components. (ReportingServices).parameters.reportingClientCertificateThumbprint.value

### Configure Service Settings

**FIELD**

- DYNAMICS 365 DNS INFORMATION - The DNS host name of the Microsoft Dynamics 365 instance, such as ax.d365ffo.onprem.contoso.com.
- AOS SERVICE PRINCIPAL USER SETTINGS - The domain user account to run the AX service, such as yourdomain\axserviceuser.
- MR SERVICE PRINCIPAL USER SETTINGS - The group managed service account (gMSA) to run the MR application service, such as yourdomain\Svc-FRAS$.
- The group managed service account (gMSA) to run the MR process service, such as yourdomain\Svc-FRPS$.

**CONFIGURATION SETTING**

- components.(AOS).parameters.infrastructure.hostName
- components. (AOS).parameters.infrastructure.principalUserAccountName *
- components. (FinancialReporting).parameters.ApplicationServicePrincipalUser.accountName.value *
- components. (FinancialReporting).parameters.ProcessServicePrincipalUser.accountName.value *
The group managed service account (gMSA) to run the MR click-once service, such as yourdomain\Svc-FRCO$.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>CONFIGURATION SETTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>The thumbprint of the Data Encryption certificate.</td>
<td>components. (AOS).parameters.database.dataEncryptionCertificateThumbprint.value</td>
</tr>
<tr>
<td>The thumbprint of the Data Signing certificate.</td>
<td>components. (AOS).parameters.database.dataSigningCertificateThumbprint.value</td>
</tr>
<tr>
<td>The thumbprint of the Session Authentication certificate.</td>
<td>components. (FinancialReporting).parameters.sessionAuthenticationCertificateThumbprint.value</td>
</tr>
<tr>
<td>The thumbprint of the SSL certificate used for WCF/SOAP support.</td>
<td>components. (AOS).parameters.infrastructure.sslCertificateThumbprint.value</td>
</tr>
<tr>
<td>The thumbprint used by the Management Reporter to communicate with AX service.</td>
<td>components. (FinancialReporting).parameters.tokenSpec.certThumbprint.value</td>
</tr>
</tbody>
</table>

**NOTE**
Remove the extra backslash from the Principal username configuration value in the .json file before entering in the LCS UI. For example, contoso\AXServiceUser should be entered as contoso\AXServiceUser in LCS.

### Application certificate settings

<table>
<thead>
<tr>
<th>FIELD</th>
<th>CONFIGURATION SETTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>The thumbprint of the Data Encryption certificate.</td>
<td>components. (AOS).parameters.database.dataEncryptionCertificateThumbprint.value</td>
</tr>
<tr>
<td>The thumbprint of the Data Signing certificate.</td>
<td>components. (AOS).parameters.database.dataSigningCertificateThumbprint.value</td>
</tr>
<tr>
<td>The thumbprint of the Session Authentication certificate.</td>
<td>components. (FinancialReporting).parameters.sessionAuthenticationCertificateThumbprint.value</td>
</tr>
<tr>
<td>The thumbprint of the SSL certificate used for WCF/SOAP support.</td>
<td>components. (AOS).parameters.infrastructure.sslCertificateThumbprint.value</td>
</tr>
<tr>
<td>The thumbprint used by the Management Reporter to communicate with AX service.</td>
<td>components. (FinancialReporting).parameters.tokenSpec.certThumbprint.value</td>
</tr>
</tbody>
</table>

### Redeploy your environment

The following instructions provide information about how to update or redeploy your environment with a new platform or topology.

1. In LCS, navigate to the **Environments** blade in your on-premises project.

2. Click **Delete** to delete your environment.

   **NOTE**
   Deleting the environment will not delete the database, infrastructure or Local agent. Only the Service Fabric applications are deleted.

3. Wait for a few minutes and verify that the deployment is deleted. To confirm the deployment is deleted, log in to the on-premises environment and navigate to the Service Fabric Explorer.

   The following applications should be deleted:
   - AXBootstapperAppType
The following on-premises service fabric agent applications should not be deleted:

- LocalAgentType
- MonitoringAgentAppType

4. After all of the applications in step 3 are deleted, go back to LCS and click **Configure**.

5. Select the new topology for your platform.

6. Enter the environment name. You can use the same name or enter a new one.

7. Click **Advanced Settings**. You can now use the relevant configurations from the .json file that you saved to configure your environment.
This topic explains how to remove an Application Object Server (AOS) node in your on-premises environment to reduce or replace a failed node. It also explains how to add a new AOS node for scale-out performance.

Remove a node

Option 1: Use a configuration file (preferred option)

Reference document: Add or remove nodes to a standalone Service Fabric cluster running on Windows Server

1. In Service Fabric Explorer, select Cluster, and make a note of the Microsoft Service Fabric cluster version. For this example, the cluster version is 6.5.676.9590.

2. On one of the orchestrator nodes, open File Explorer. On the View tab, in the Show/hide group, make sure that the File name extensions and Hidden items check boxes are selected.

3. Expand drive C, and then drill down into the following folder. (Note that the bold parts of the path will vary, depending on the node name and setup.)

C:\ProgramData\SF\ORCH1\Fabric\work\Applications\__FabricSystem\_App4294967295\work\Store\131811633624852852

In the folder, you should see a list of folders for various versions of Microsoft Service Fabric. Here is an example.
4. Open the folder with the name the same as the version of Microsoft Service Fabric cluster you that you made a note of earlier. For this example, the folder is named 6.5.676.9590.

5. In the folder, you should see a .cab file.

6. Copy the .cab file to C:\Temp, and rename the copied file MicrosoftAzureServiceFabric.cab. (If you don't have a Temp folder, create it.)

7. Open a Windows PowerShell Command Prompt window as an admin.

8. Run the following command to connect to the Service Fabric cluster.

```powershell
#Connect to Service Fabric Cluster. Replace 123 with server/star thumbprint and use appropriate IP address
Connect-ServiceFabricCluster -connectionEndpoint 10.0.0.12:19000 -X509Credential -FindType FindByThumbprint -FindValue 123 -ServerCertThumbprint 123
```
9. Run the following command to save the configuration file to C:\Temp\ClusterConfig.json. (Make sure that the C:\Temp path exists.)

```
Get-ServiceFabricClusterConfiguration > C:\Temp\ClusterConfig.json
```

10. In the configuration file that you saved in the previous step, in the `fabricSettings` section, in the `Setup` section, add a section for the `NodesToBeRemoved` parameter. The parameter value should be a comma-separated list of names of the nodes that must be removed.

```
"fabricSettings": [
  {
    "name": "Setup",
    "parameters": [
      {
        "name": "FabricDataRoot",
        "value": "C:\\ProgramData\\SF"
      },
      {
        "name": "FabricLogRoot",
        "value": "C:\\ProgramData\\SF\\Log"
      },
      {
        "name": "NodesToBeRemoved",
        "value": "AOS1"
      }
    ]
  }
]
```

11. Remove the node from the `Nodes` section. In the following example, the `AOS1` node was removed.
"Nodes": [  
  
  {  
      "NodeName": "AOS2",  
      "NodeTypeRef": "AOSNodeType",  
      "IPAddress": "10.0.0.10",  
      "FaultDomain": "fd:/fd1",  
      "UpgradeDomain": "ud1"  
  },  
  
  {  
      "NodeName": "AOS3",  
      "NodeTypeRef": "AOSNo...  
  }
]

Start-ServiceFabricClusterConfigurationUpgrade -ClusterConfigPath C:\Temp\ClusterConfig.json

Get-ServiceFabricClusterUpgrade

12. Remove the following lines from the Security section.

"WindowsIdentities": {  
  "\$id": "3"  
}

NOTE
If you don't remove these lines, you will receive the following error message later:

ValidationException: Authentication type can't be changed from unsecured to Windows.

13. Increment the version number of the configuration file. Make this change at the lowest increment. In the following example, the version number went from 1.0.0 to 1.0.1.

"ClusterConfigurationVersion": "1.0.1"

14. Save the configuration file.

15. Run the following command to remove the node.

Start-ServiceFabricClusterConfigurationUpgrade -ClusterConfigPath C:\Temp\ClusterConfig.json

16. Run the following command to monitor the progress.

Get-ServiceFabricClusterUpgrade

If the upgrade stops responding at "UpgradePhase: PreUpgradeSafetyCheck," make a note of the NodeName value, and restart that node from Service Fabric Explorer. In the following illustration, the upgrade has stopped responding. It was running for 50 minutes at the same status on node BI1.
During upgrade of the cluster configuration, if you receive an error message that states that you previously added a node through the `Add-ServiceFabricNode` command, you will need to run a configuration upgrade without making any changes to the configuration file except for the version number. You can use the `Get-ServiceFabricClusterConfiguration` and `Start-ServiceFabricClusterConfigurationUpgrade` commands for this purpose.

You can also view the progress in Service Fabric Explorer.

**Option 2: Use Service Fabric Explorer**

1. Sign in to Service Fabric Explorer.
2. Select the **Settings** button (gear symbol), and make sure that **Advanced** mode is turned on.
3. Expand **Nodes**, select the ellipsis (…) button next to the node that you want to remove, and then select **Deactivate (remove data)**. Note that this option might not be available if the node is already down (for example, if the node server can't be started).

4. When you're prompted to confirm deactivation, enter the name of the node, and then select **Deactivate (remove data)**.

![Confirm Node Deactivation](image)

After the node has been deactivated, its status is shown as **Disabled**.
5. If the server is still active and connected to the domain, you might have to follow these steps if you will be replacing the deactivated node with a new server:
   
   a. Sign in to the server.
   b. Remove the server from the domain.
   c. Rename the server.
   d. Make a note of the IP address, and then change the IP address to a free address that you have in your range.
   e. Shut down the server.

6. After the server has been shut down, or if it was already down, Service Fabric Explorer reflects its status. Select the ellipsis (…) button again next to the node, and then select **Remove node state**.

7. Confirm removal of the node.

8. Make a note of the node name and type. For this example, the node name is **AOS1**, and the type is **AOSNodeType**. Remember that the node name might not match the network name. Also make a note of the **Upgrade Domain** and **Fault Domain** settings, and the IP address. The previous illustration shows all these values.
Add a node

The next step is to start a new AOS server.

1. Follow these steps if you’re replacing an existing server that was removed:
   a. Give the server the network name of the previous AOS server.
   b. Assign the original IP address. For this example, that IP address is 10.0.0.9.
   c. Join the server to the domain.

2. If you’re adding a new server to an existing cluster, update the ConfigTemplate.xml file so that it contains the additional information. This information will be used when you push out the prerequisites and apply settings through Windows PowerShell scripts.

3. Make sure that you’ve added the AXServiceUser and svc-AXSF$ group Managed Service Accounts (gMSAs) to the local admin group on the AOS server.

After the server is connected to the domain, you must follow the prerequisite steps for on-premises environments in Set up and deploy on-premises environments (Platform update 12 and later). The following steps summarize those prerequisite steps.

4. Copy the contents of each infrastructure\VMs<VMName> folder into the corresponding virtual machine (VM). (If you use remoting scripts, they will automatically copy the contents to the target VMs.) Then run the following Windows PowerShell scripts as an admin.

   ![NOTE]
   If you’re running remotely and repairing an existing server, specify the -ForcePushLBDScripts switch to ensure the file copy process is run against all servers again.

```
# Install pre-req software on the VMs.
# If Remoting, execute
# .\Configure-PreReqs-AllVMs.ps1 -MSIFilePath <share folder path of the MSIs> -ConfigurationFilePath .\ConfigTemplate.xml -ForcePushLBDScripts
# .\Configure-PreReqs.ps1 -MSIFilePath <share folder path of the MSIs>
```

5. Restart the computer every time that you’re prompted. Make sure that you rerun the .\Configure-PreReqs.ps1 script after every restart, until all the prerequisites are installed. In the case of remoting, rerun the AllVMs script when all the computers are back online.

6. If you use the remoting script, make sure that the current user has access to the share folder of Microsoft Windows Installer package files (.msi files).

7. If you use the remoting script, make sure that no user is accessing computers of the AOSNodeType, MRType, and ReportServerType types. Otherwise, the remoting script won’t be able to restart the computer because users are signed in to it.

8. Run the following scripts, if they exist, to complete the VM setup.

```
# If Remoting, only execute
# .\Complete-PreReqs-AllVMs.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
# .\Add-GMSAonVM.ps1
# .\Import-PfxFiles.ps1
# .\Set-CertificateAcls.ps1
```

9. If errors occur while you run Add-GMSAonVM.ps1, you likely need to update your gMSA account. Run the following script from your infrastructure scripts folder.
10. Run the following script to validate the VM setup.

    # If Remoting, execute
    # .\Test-D365FOConfiguration-AllVMs.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
    .\Test-D365FOConfiguration.ps1

11. Before you continue, fix anything that fails as part of the validation script.

    For this example, the cluster version is 6.5.676.9590.

13. On one of the orchestrator nodes, open File Explorer. On the View tab, in the Show/hide group, make sure that the File name extensions and Hidden items check boxes are selected.

14. Expand drive C, and then drill down into the following folder. (Note that the bold parts of the path will vary, depending on the node name and setup.)

    C:\ProgramData\SF\ORCH1\Fabric\work\Applications\__FabricSystem\_App4294967295\work\Store\131811633624852852

    In the folder, you should see a list of folders for various versions of Service Fabric. Here is an example.
15. Open the folder that has the same name as the version of Microsoft Service Fabric cluster that you made a note of earlier. For this example, the folder is named 6.5.676.9590.

16. In the folder, you should see a .cab file.

17. Copy the .cab file to C:\Temp, and rename the copied file MicrosoftAzureServiceFabric.cab. (If you don't have a Temp folder, create it.)

18. Open a Windows PowerShell Command Prompt windows as an admin.

19. Run the following command to connect to your Service Fabric cluster. (Edit the command as you require.)

```powershell
#Connect to Service Fabric Cluster. Replace 123 with server/star thumbprint and use appropriate IP address
Connect-ServiceFabricCluster -connectionEndpoint 10.0.0.12:19000 -X509Credential -FindType FindByThumbprint -FindValue 123 -ServerCertThumbprint 123
```
20. Run the following command to add the node back in. Before you run it, make the required edits to the
NodeName, IPAddress, UpgradeDomain, and FaultDomain parameters. (If you're replacing an
existing server, you should have made a note of the values earlier.)

```
Add-ServiceFabricNode -NodeName "AOS1" -NodeType "AOSNodeType" -IpAddressOrFQDN "10.0.0.9" -
UpgradeDomain "ud0" -FaultDomain "fd:/fd0" -FabricRuntimePackagePath
"C:\Temp\MicrosoftAzureServiceFabric.cab"
```

21. After the node has been added back in, return to Service Fabric Explorer, and view the application
deployment status. Several minutes will be required before all the AOS applications are restored
(AXBootstrapperAppType, AXSFType, RTGatewayAppType, and LBDTelemetryType-<envname>
or MonitoringAgentAppType) are pushed out again and installed on the node.
Reuse the same AD FS instance for multiple environments

11/24/2021 • 2 minutes to read • Edit Online

This topic explains how to use the same instance of Active Directory Federation Services (AD FS) in multiple Microsoft Dynamics 365 Finance + Operations (on-premises) environments.

Setup

**IMPORTANT**
This procedure assumes that you've previously configured AD FS for one environment by following the instructions in the [Set up and deploy on-premises environments](#) content. It also assumes that that environment is running without any issues.

1. In AD FS Manager, go to AD FS > Application groups, and open Microsoft Dynamics 365 for Operations On-premises.

2. In the Native application section, follow these steps:

3. In the Web API section, follow these steps:

4. Optional: In the Server section, open Microsoft Dynamics 365 for Operations On-premises - Retail, and add the redirect URI of the new environment (https://ax.contoso.com/namespaces/AXSF/).

5. Optional: Configure the warehouse mobile app for the new environment by following the instructions in [Configure the Warehousing app for on-premises deployments](#) again. Remember to use the URI of the new environment (https://ax.contoso.com) as the Resource URL value.

**NOTE**
No additional configuration is required for the workflow and retail designer applications.

6. Verify that you can reach the OpenID metadata endpoint (https://<adfs-dns-name>/adfs/.well-known/openid-configuration) from the AOS and MR nodes in your new environment. If you're using self-signed certificates, you might have to import the AD FS Secure Sockets Layer (SSL) certificate into the Trusted Root Certification Authorities store of each node.

7. When you deploy the new environment from Microsoft Dynamics Lifecycle Services (LCS) and are
specifying the deployment configuration, make sure that you use the same AD FS OpenID metadata endpoint and AD FS OpenID connect client IDs that you specified for the previous environment.
This topic explains how to use the same instance of Active Directory Federation Services (AD FS) for a Dynamics 365 Finance + Operations (on-premises) environment and for Microsoft 365.

Existing deployments

1. Download the new local agent version from Microsoft Dynamics Lifecycle Services (LCS). It should be version 2.2.0 or later.

2. Download the new version of the local agent configuration file, because it has additional configuration that is required for this functionality.

3. Modify the new local agent configuration file, and set the `office365AdfsCompatibility` value to `True`.

4. Run the following command to uninstall the old local agent version from your cluster.

   ```bash
   .\LocalAgentCLI.exe Cleanup '<path of localagent-config.json>'
   ```

5. Run the following command to install the new local agent version.

   ```bash
   .\LocalAgentCLI.exe Install '<path of localagent-config.json>'
   ```

6. Perform any servicing operation with Platform update 28 or later to make the new configuration available.

7. After servicing is completed, run the following script.

   ```bash
   .\Reset-DatabaseUsers.ps1 -DatabaseServer '<FQDN of the SQL server>' -DatabaseName '<AX database name>'
   ```

   **IMPORTANT**
   
   If you skip this step, the primary admin user won't be able to sign in.

8. Use Service Fabric Explorer to **Restart applications (such as AOS)**.

9. Verify that you are able to sign in to the product with the system administrator user that was specified during deployment.

10. Download the newest version of the infrastructure scripts from the LCS Shared asset library.

11. Copy the `Reset-SID.ps1` script from the downloaded infrastructure scripts folder into one of your AOS machines.
12. Execute the `Reset-Sid.ps1` script.

   ```powershell
   .\Reset-Sid.ps1 -AxsfCodePath "C:\ProgramData\SF\AOS_13\Fabric\work\Applications\AXSFType_App184\AXSF.Code.1.0.20190902"
   ```

New deployments

1. Follow the instructions for installing the local agent in the "Configure a connector and install an on-premises local agent" section of Set up and deploy on-premises environments. However, before you actually install the local agent, complete step 2 of this procedure.

2. Modify the local agent configuration file, and set the `office365AdfsCompatibility` value to `True`.

3. Continue to follow the instructions in the "Configure a connector and install an on-premises local agent" section of Set up and deploy on-premises environments, and deploy a base version that runs Platform update 28 or later. If there is no base version that runs Platform update 28 or later, deploy the latest base version that is available. Then service it so that Platform update 28 is deployed on top.

Partial support

For partial support, it is necessary to have Local Agent 2.2.0 or later installed and to update the service with Platform update 28 or later.

With partial support, authentication against the Financial Reporting service is not supported.
Local agent 2.3.0 supports the execution of pre-deployment and post-deployment scripts. Therefore, customers can now set up Microsoft Windows PowerShell scripts that are run before and after the environment is deployed. This feature applies to deployments and redeployments, and also to servicing operations.

To make this feature available, you must create a Scripts folder in the agent file share. To run a pre-deployment script, create a `PreDeployment.ps1` file in the Scripts folder. To run a post-deployment script, create a `PostDeployment.ps1` file. The following examples show the folder structure:

- `\\fileserver\agent\scripts\PreDeployment.ps1`
- `\\fileserver\agent\scripts\PostDeployment.ps1`

If these files don't exist, the deployment continues as usual. The following examples show the new deployment flow.

**Deployment or redeployment:**

1. Get unhealthy modules. In this step, the health of existing services is obtained to find which are unhealthy. This step applies only to redeployment scenarios.
2. Clean up modules. In this step, the services are removed and the contents of the wp folder are deleted. This step applies only to redeployment scenarios.
3. Link download artifacts. In this step, download, extraction, and processing of artifacts from Microsoft Dynamics Lifecycle Services (LCS) takes place.
4. Pre-deployment script. In this step the `PreDeployment.ps1` script is executed (if it exists).
5. Set up modules. In this step, the new services are deployed.
6. Post-Deployment script. In this step the `PostDeployment.ps1` script is executed (if it exists).

**Servicing:**

1. Prepare for servicing. In this step, the package is prepared in LCS and gets downloaded to the environment.
2. Clean up modules. In this step, the services are removed and the contents of the wp folder are deleted.
3. Link download artifacts. In this step, extraction and processing of previously downloaded artifacts from LCS takes place.
4. Pre-deployment script. In this step the `PreDeployment.ps1` script is executed (if it exists).
5. Set up modules. In this step, the new services are deployed.
6. Post-Deployment script. In this step the `PostDeployment.ps1` script is executed (if it exists).

**NOTE**

The pre-deployment and post-deployment scripts can contain anything. The code that the scripts run is solely the customer's responsibility. The local agent just invokes the scripts.

**Customizations**

The default time-out for script execution is 30 minutes. To change this value, modify the localagent-config.json file, and then reinstall the local agent by using the modified file. The following attribute defines time-out value
"powershellScriptRunner": {
    "timeoutMinutes": {
        "value": "30"
    }
}

Logging

The outputs and error messages from the scripts are written, as .log and .err files, into the Logs folder in the Scripts folder. If a script times out, only an error message is logged. This error message has a time-out message. No other outputs are logged in this situation.

Execution of the scripts is also logged as Event Tracing for Windows (ETW) events. You can view these events in Event Viewer. If a script produces any errors, an error event is logged, but deployment continues as usual.
This topic explains which deployment configurations can be specified, when deploying the local agent, to indicate a special configuration related to the environment.

There is a section in the localagent-config.json file that is labeled deploymentOptions. This can be modified before installing the local agent.

```
...
"deploymentOptions": {
  "office365AdfsCompatibility": {
    "value": "false"
  },
  "sqlServerVersion": {
    "value": "2016"
  },
  "isMultiSubnetFailoverEnabled": {
    "value": "false"
  },
  "skipCRLCheck": {
    "value": "false"
  }
},
...
```

Specify the version of Microsoft SQL Server

If your environment has Microsoft SQL Server 2019 installed throughout the different components, change `sqlServerVersion` from the default of 2016, to 2019.

For a list of compatible SQL Server versions, see [Microsoft Dynamics 365 Finance + Operations (on-premises) supported software](#).

Specify that AD FS is deployed with Microsoft 365 compatibility

To specify that Active Directory Federation Services (AD FS) is deployed with Microsoft 365 compatibility, change `office365AdfsCompatibility` from `false` to `true`.

For more information, see [AD FS Microsoft 365 compatibility](#).

Specify that the SQL Server cluster is deployed in multiple subnets

To specify that the SQL Server cluster is deployed in multiple subnets, change `isMultiSubnetFailoverEnabled` from `false` to `true`.

For more information on this SQL Server configuration, see [SQL Server Multi-Subnet Clustering](#).

Support for this feature was introduced in release 10.0.19.

Specify that checking the certificate revocation list of a certificate should be skipped

As part of establishing a trusted connection between a client and a server, one of the checks that is carried out is
checking that the certificate provided by the server has not been revoked by the issuing authority.

This requires that a client, such as the FinancialReporting service, retrieve the certificate revocation list. If the certificate has been issued by a public certificate authority, then the client would need access to the internet in order to verify that the certificate has not been revoked.

Some on-premises environments are not allowed to connect to the internet. As such, they may not be able to perform this check. It is possible to disable this check by updating `skipCRLCheck` from `false` to `true`.

Support for this option was introduced in version 10.0.21. Additionally, at least local agent 2.7.1 is required to use this option.

**IMPORTANT**

By disabling the certificate revocation list of a certificate, the security check will not be performed. You bear the risk of disabling it. You should only enable this deployment option if you are fully aware of the security implications of disabling this check.
This topic explains how to update the local agent. The latest version of the local agent is version 3.0.0, which was released in November 2021.

<table>
<thead>
<tr>
<th>LOCAL AGENT VERSION</th>
<th>CAPABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.0.0</td>
<td>This version includes support for Edge Scale Unit Application Lifecycle Management.</td>
</tr>
<tr>
<td>2.7.2</td>
<td>This version includes a fix for deploying older application versions.</td>
</tr>
<tr>
<td>2.7.1</td>
<td>This version introduces a new deployment option and fixes a bug with a deployment option.</td>
</tr>
<tr>
<td>2.7.0</td>
<td>Enables deploying or updating to 10.0.21 and later versions. Additionally, this version enables deploying on environments with Microsoft SQL Server 2019 and some bug fixes.</td>
</tr>
<tr>
<td>2.6.0</td>
<td>This version upgrades the Service Fabric SDK, fixes a bug with refresh state, and increases the application provisioning timeout.</td>
</tr>
<tr>
<td>2.5.0</td>
<td>This version updates dependencies and fixes a cleanup bug.</td>
</tr>
<tr>
<td>2.4.0</td>
<td>This version fixes a deployment issue and upgrades the runtime of the local agent.</td>
</tr>
</tbody>
</table>
| 2.3.1               | This version fixes orchestration service crashes that may occur during clean up on some environments. 
Deploying version 10.0.5 with Platform update 29 or earlier requires the use of pre-deployment scripts for automatic updating of FinancialReportingDeployer.exe.config. For more information, see Troubleshoot on-premises deployments. |
<p>| 2.3.0               | This version adds support for pre- and post-deployment scripts. |
| 2.2.0               | This version fixes locked dlls during cleanup and enables prerequisites for supporting Active Directory Federation Services (AD FS) that also is used for Microsoft 365. |
| 2.1.2               | This version contains updated Azure dependencies for improved download stability and logic to correctly evaluate if files are downloaded. This fixes an issue where files are fully downloaded, but the logic would still consider them as missing a few bytes and therefore fail the download. |</p>
<table>
<thead>
<tr>
<th>LOCAL AGENT VERSION</th>
<th>CAPABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.1</td>
<td>This version fixes an issue that occurs when the download fails and the Lifecycle Services (LCS) <strong>Maintain</strong> button is not available. Additional changes include updates to Azure storage libraries to improve communication with Azure storage and enable TLS 1.2.</td>
</tr>
<tr>
<td>2.1.0</td>
<td>This version enables two-phased servicing where <strong>Preparation</strong> and <strong>Update</strong> are two separate steps.</td>
</tr>
<tr>
<td>2.0.0</td>
<td>This version enables servicing flows and deploys Platform update 12.</td>
</tr>
<tr>
<td>1.1.0</td>
<td>This version enables the <strong>Reconfigure feature</strong> for successful deployments, enables multi-model package deployments, and deploys Platform update 8 and 11.</td>
</tr>
<tr>
<td>1.0.0</td>
<td>This version enables the <strong>Reconfigure feature</strong> for failed deployments.</td>
</tr>
<tr>
<td>Null</td>
<td>This initial version deploys Platform update 8.</td>
</tr>
</tbody>
</table>

**What's new in local agent 3.0.0**

- Local agent 3.0.0 includes support for managing the lifecycle of Edge Scale Units through the Scale Unit Management portal. For more information, see [Distributed Hybrid Topology](#).
- This release requires the .NET Framework version 4.8 to uptake the newest changes from LCS.

**What's new in local agent 2.7.2**

- Local agent 2.7.2 fixes an issue where environments on older versions of the application would fail to deploy.

**What's new in local agent 2.7.1**

- Local agent 2.7.1 introduces a new deployment option to **Specify that checking the Certificate Revocation List of a certificate should be skipped**.
- This release addresses a bug where the `office365AdfsCompatibility` deploymentOption was not being correctly set.

**What's new in local agent 2.7.0**

- Local agent 2.7.0 is a prerequisite to deploy or update to 10.0.21 and later releases.
- This release introduces the possibility of specifying a limited set of deployment options for environment-specific deployment options. Most notably, this release will allow you to deploy on environments with Microsoft SQL Server 2019. For all possible configurations, see [Local agent deployment configurations](#).
- Additionally, this release addresses an issue where the gMSA account that the local agent executes under will lose permission to the private key for some certificates.
- The LBDTelemetry-Agent application can start correctly even if the Event Viewer is open.
What's new in local agent 2.6.0

- Local agent 2.6.0 uptakes a new Service Fabric SDK and runtime.
- This release fixes a bug where, if refresh state is triggered when the environment is stuck in the Downloading phase, the environment would automatically move to a deployed state without updating the environment. In this situation, the refresh state will mark the Downloading phase as failed.
- The timeout for provisioning an application has been increased.

What's new in local agent 2.5.0

- Local agent 2.5.0 uptakes new versions of various dependencies. The main changes are Service Fabric and Entity Framework.
- This release also fixes a bug where, if cleanup fails without cleaning up any services, subsequent reattempts always fail during cleanup.

What's new in local agent 2.4.0

- Local agent 2.4.0 now requires the .NET Framework version 4.7.2 to uptake the newest changes from LCS. To meet the newest requirements, be sure to run the latest infrastructure scripts that are available in LCS.
- This release also fixes an issue where the deployment of the AXService would fail in slower environments due to a hard-coded timeout.

What's new in local agent 2.3.0

- Local agent 2.3.0 enables the execution of custom pre- and post- deployment scripts.
- It fixes the problem introduced in 2.2.0 with regard to deploying older platform updates.
- This release removes the monitoring agent and introduces a new service called LBDTelemetry, which will be used to install the ETWManifests.

What's new in local agent 2.1.0

- Local agent 2.1.0 enables the two-phased servicing where Environment preparation and Environment update are two distinct steps and explicit actions. This reduces the total downtime customers must take when applying updates to their on-premises environments by preparing upfront and allowing users to use the environment during preparation and then communicating the downtime when the actual update environment action is triggered.
What's new in local agent 2.0.0

- Local agent 2.0.0 can deploy Platform update 12.
- It enables the Reconfigure feature until the first deployment of Platform update 12 succeeds.
- It disables the Reconfigure feature on the first successful deployment of Platform update 12. After deployment succeeds, you can use the regular update experience to update the environment.

**NOTE**
Local agent 2.0.0 cannot deploy Platform update 8 and Platform update 11. You must have version 1.1.0 to deploy those platform updates.

Download the latest local agent and configuration from LCS

**NOTE**
If you require an older version of the local agent for your current deployments, download it from the Asset library in Microsoft Dynamics Lifecycle Services (LCS). To download Local agent version 1.1.0, go to **Shared Asset Library -> Model** and click on Dynamics 365 for Finance and Operations on-premises - Local agent v1.1.0**. You must have version 2.0.0 or later to deploy Platform update 12 and complete update flows.

1. In LCS, select **Project settings > On-prem connectors**.
2. Select the connector to your environment, and then select **Edit**.
3. On the menu on the left side of the page, select **Setup host infrastructure**, and then select **Download agent installer**.
   
   You must now verify that the zip file that is downloaded and unblocked.
4. Go to the zip file, right-click it, and then select **Properties**.
5. In the **Properties** dialog box, select **Unblock**, and then select **Apply**.
6. On the **Configure agent** tab, select **Download configurations** to download the localagent-config.json configuration file.

Update the local agent

1. Copy the zip file and the localagent-config.json file into one of the **Orchestrator** nodes, such as `c:\DynamicsAgent` in the **Orch1** virtual machine (VM).
2. Unzip the agent installer to `C:\DynamicsAgent\LocalAgent`.
3. Copy the localagent-config.json file to `C:\DynamicsAgent\LocalAgent`.
4. In a **Command Prompt** window, go to `C:\DynamicsAgent\LocalAgent`, and run the following command.

```
LocalAgentCLI.exe Cleanup <path of localagent-config.json>
```

**NOTE**
You must use the current agent's binaries to clean up the agent. If you don't have the current agent's binaries, you can delete the local agent application from Service Fabric Explorer.
5. Press any key to exit the cleanup operation.

6. Verify that the local agent has been successfully cleaned up by looking in Service Fabric Explorer and making sure that there are no apps in the **Deployed Applications** section in the **Orchestrator** nodes.

7. After the local agent is successfully cleaned up, run the following command.

```
LocalAgentCLI.exe Install <path of localagent-config.json>
```

8. After the local agent is successfully installed, go back to your on-premises connector in LCS.

9. On the **Validate setup** tab, select **Message agent** to test LCS connectivity to your new local agent.
This topic provides troubleshooting information for deployments of Microsoft Dynamics 365 Finance + Operations (on-premises).

Access Service Fabric Explorer

You can access Service Fabric Explorer in a web browser by using the default address, https://sf.d365ffo.onprem.contoso.com:19080. To verify the address, note the value that was used in the "Create DNS zones and add A records" section of the appropriate setup and deployment topic for your environment:

- Platform update 41 and later
- Platform updates 12 through 40

You can access the site only if the client certificate is in cert:\CurrentUser\My on the machine that you're accessing the site on. (In Certificate Manager, go to Certificates - Current User - Personal - Certificates.) When you access the site, select the client certificate when you're prompted.

Monitor the deployment

Identify the primary orchestrator

To determine the machine that is the primary instance for stateful services such as a local agent, in Service Fabric Explorer, expand Cluster > Applications > <intended application example> LocalAgentType > fabric:/LocalAgent/OrchestrationService > (GUID).

The primary node is shown. For stateless services or the remaining applications, you must check all the nodes.

Note the following points:

- OrchestrationService orchestrates the deployment and servicing actions for Finance + Operations.
- ArtifactsManager downloads files from Microsoft Azure cloud storage to the local agent file share. It also unzips the files into the required format.

Review the orchestrator event logs

From the primary OrchestrationService orchestrator machine, in Event Viewer, go to Applications and Services Logs > Microsoft > Dynamics > AX-LocalAgent.

NOTE

To view the full error message, you must select the Details tab.

The following modules must be installed:

- Common
- ReportingServices
- AOS
- FinancialReporting
The following commands must be run:

- **Setup**
- **Dvt** – This command runs a deployment verification test.
- **Cleanup** – This command is used to service and delete an environment.

The following folders contain additional information:

- AX-SetupModuleEvents
- AX-SetupInfrastructureEvents
- AX-BridgeService

To review Microsoft Dynamics entries in Event Viewer, follow these steps.

1. In Event Viewer, right-click **Custom Views**, and then select **Create Custom View**.
2. In the **Event logs** field, select **Dynamics**.

   ![Create Custom View](image)

   **NOTE**
   Also look at **Administrative Events** in **Custom Views**.

**Service Fabric Explorer**

Note the state of the cluster, application, and nodes. For information about how to access Service Fabric Explorer, see **Access Service Fabric Explorer**.

**Error:** "Partition is below target replica or instance count"

You might receive the following error:

```
Partition is below target replica or instance count
```

This error isn't a root error. It indicates that the status of each node isn't ready. For AXSFType (AOS), the status might still be **InBuild**.
On the machines that are related to the error message, use Event Viewer to view the latest activity.

**AXSFType**

If a status of **InBuild** is shown for AXSFType (AOS), review the DB Sync status and other events from Application Object Server (AOS) machines.

To diagnose errors, use Event Viewer to review the following event logs:

- Applications and Services Logs > Microsoft > Dynamics > AX-DatabaseSynchronize
- Custom Views > Administrative Events

**Error: “ExtractInstallerService failed to extract”**

C:\Users\Dynuser\CONTOSO\AppData\Local\Temp\1blssblh.w0n\FabricInstallerService.Code\FabricClient.dll

You might receive the following error:

"ExtractInstallerService failed to extract"
C:\Users\Dynuser\CONTOSO\AppData\Local\Temp\1blssblh.w0n\FabricInstallerService.Code\FabricClient.dll.

If you receive this error, download the latest version of **Azure Service Fabric**. Note that the user name and path in the error message vary, depending on your environment.

**Service Fabric logs**

You can find more details about Service Fabric applications in the log files at C:\ProgramData\SF\<OrchestratorMachineName>\Fabric\work\Applications\LocalAgentType_App<N>\log.

**Lifecycle Services**

Note the current deployment status for the environment in Microsoft Dynamics Lifecycle Services (LCS).

**A time-out error occurs when a Service Fabric cluster is created**

Run Test-D365FOConfiguration.ps1 as noted in the “Set up a standalone Service Fabric cluster” section of the appropriate setup and deployment topic for your environment. Note any errors.

- **Platform update 41 and later**
- **Platform updates 12 through 40**

Be sure to complete these steps:

- Verify that the Service Fabric Server client certificate exists in the LocalMachine store on all Service Fabric nodes.
- Verify that the Service Fabric Server certificate has the access control list (ACL) for Network Service on all Service Fabric nodes.
- Review the antivirus exclusions that are noted in **Environment setup**.

**A time-out error occurs while you're waiting for Installer Service to be completed for machine x.x.x.x**

Only one node type is supported for each Internet Protocol (IP) address (that is, for each machine). Check whether the nodes are being reused on the same machine. For example, AOS and ORCH must not be on the same machine, and ConfigTemplate.xml must be correctly defined.

**Remove a specific application**

We recommend that you use LCS to remove or clean up deployments. However, you can also use Service Fabric Explorer to remove an application as you require.
In Service Fabric Explorer, go to **Application node > Applications > MonitoringAgentAppType-Agent.** Select the ellipsis button (…) next to `fabric:/Agent-Monitoring`, and delete the application. Enter the full name of the application to confirm the deletion of the application.

You can also remove `MonitoringAgentAppType-Agent` by selecting the ellipsis button and then selecting **Unprovision Type.** Enter the full name to confirm the removal of the application.

### Remove all applications from Service Fabric

The following script removes and unprovisions all Service Fabric applications except LocalAgent and the monitoring agent for LocalAgent. You must run this script on an orchestrator virtual machine (VM).

```
$applicationNamesToIgnore = @('fabric:/LocalAgent', 'fabric:/Agent-Monitoring', 'fabric:/Agent-LBDTelemetry')
$applicationTypeNamesToIgnore = @('MonitoringAgentAppType-Agent', 'LocalAgentType', 'LBDTelemetryType-Agent')

Get-ServiceFabricApplication | ` Where-Object { $_.ApplicationName -notin $applicationNamesToIgnore } | ` Remove-ServiceFabricApplication -Force

Get-ServiceFabricApplicationType | ` Where-Object { $_.ApplicationTypeName -notin $applicationTypeNamesToIgnore } | ` Unregister-ServiceFabricApplicationType -Force
```

### Remove Service Fabric

To completely remove the Service Fabric cluster, follow these steps.

1. Run the following command.

   ```
   .\RemoveServiceFabricCluster.ps1 -ClusterConfigFilePath .\ClusterConfig.json
   ```

2. If an error occurs, remove a specific node on the cluster by using the **CleanFabric.ps1** command. You can find this command in `C:\Program Files\Microsoft Service Fabric\bin\fabric\fabric.code`.

3. Remove the `C:\ProgramData\SF` folder, if you're using the default location. Otherwise, remove the specified folder.
   
   If you receive an "Access denied" error, restart Microsoft Windows PowerShell or the machine.

### Clean up an existing environment and redeploy

To clean up an existing environment and redeploy, follow these steps.

1. In LCS, open the project, and then, in the **Environments** section, delete the deployment.

   The applications should start to disappear from Service Fabric Explorer in the environment. This process will take one to two minutes.

2. Access the orchestrator machine that contains `LocalAgentCLI.exe`, and follow these steps:

   a. Run the local agent cleanup.

      ```
      .\LocalAgentCLI.exe Cleanup '<path of localagent-config.json>'
      ```

   b. Remove Service Fabric.
c. If any nodes fail, run the `CleanFabric.ps1` command. You can find this command in `C:\Program Files\Microsoft Service Fabric\bin\fabric\fabric.code`.

d. Remove the `C:\ProgramData\SF` folder on all Service Fabric nodes.
   If you receive an "Access denied" error, restart the machine, and try again.

3. Remove or update certificates as required.

   Remove old certificates from all AOS, BI, ORCH, and DC nodes.
   - The certificates exist in the following certificate stores: Cert:\CurrentUser\My, Cert:\LocalMachine\My, and Cert:\LocalMachine\Root.
   - If the setup of Microsoft SQL Server will be modified, remove the SQL Server certificates.
   - If the settings for Active Directory Federation Services (AD FS) will be modified, remove the AD FS certificate.

4. Update the following configuration files as required:
   - `ConfigTemplate.xml`
   - `ClusterConfig.json`
   For information about how to correctly fill in the fields in the templates, see the appropriate setup and deployment topic for your environment:
     - Platform update 41 and later
     - Platform updates 12 through 40

5. In LCS, open the project, and update the LCS on-premises connector as required.
   a. Re-create the LCS on-premises connector for the environment, or edit the settings of an existing connector.
      To obtain easy-to-copy values for LCS, use the `.\Get-AgentConfiguration.ps1` script.
   b. Download the latest local agent configuration, `localagent-config.json`.

6. Deploy again by following the instructions in the appropriate setup and deployment topic for the environment:
   - Platform update 41 and later
   - Platform updates 12 through 40

Find the local agent values that are used

You can find local agent values in Service Fabric Explorer. Go to `Cluster > Applications > LocalAgentType > fabric:/LocalAgent`, and then select `Details`.

Alternatively, run the following Windows PowerShell command.

```powershell
.\Get-AgentConfiguration.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
```

Install, upgrade, or uninstall a local agent

For information about how to update the local agent, see Update the local agent.

You can also use the following upgrade and uninstallation commands:
An error occurs when local agent services are started

When local agent services are started, you might receive the following error:

```
Could not load file or assembly 'Lcs.DeploymentAgent.Proxy.Contract, Version=1.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35' or one of its dependencies.
```

This error means that strong name verification is turned on. You can turn off this verification by using Configure-PreReqs.ps1. To validate that strong name verification is no longer turned on, run Test-D365FOConfiguration.ps1.

A "Validation in progress" message is shown for several minutes in LCS

Follow these steps to troubleshoot general issues with local agent validation.

1. Run `Configure-PreReqs.ps1` on all orchestrator machines to configure the machines correctly.
2. Verify that the Test-D365FOConfiguration.ps1 script passes on all the orchestrator machines.
3. Verify that the installation of LocalAgentCLI.exe is successfully completed.
4. In Service Fabric Explorer, verify that all the applications are healthy.
5. If the applications aren’t healthy, find the primary node for the service that is failing. In Event Viewer, look for events in the following locations:
   - Custom Views > Administrative Events
   - Applications and Services Log > Microsoft > Dynamics > AX-LocalAgent

Local agent errors

**Issue**

**Error:** When you install the local agent, you receive the following error.

```
LocalAgentCLI.exe Error: 0 : Exception System.InvalidOperationException: unable to get settings for telemetry setup component
  at LBDTelemetryCommon.LBDTelemetrySetupManager.GetComponentSettings()
  at LBDTelemetryCommon.LBDTelemetrySetupManager.ApplyParameters()
  at LocalAgentCLI.Program.Main(String[] args)
Press any key to exit
```

**Reason:** You're trying to install local agent version 2.3.0 or later, but the localagent-config.json file that you're using isn't up to date.

**Steps:** Get the new version of the localagent-config.json file from LCS by following the instructions in the "Configure a connector and install an on-premises local agent" section of Set up and deploy on-premises.
environments.

You can also manually add the following values in the components section of the localagent-config.json file.

```json
{
    "name": "LBDTelemetry",
    "placementCriteria": "(IsOrchestratorEnabled == True)",
    "parameters": {
        "applicationPackagePath": {
            "value": "Applications\LBDTelemetry"
        }
    }
}
```

**Issue**

**Error:** You might receive the following errors:

- Unable to process commands
- Unable to get the channel information
- RunAsync failed due to an unhandled exception causing the host process to crash:

  System.ArgumentNullException: Value cannot be null. Parameter name: certificate

**Reason:** These errors can occur because the certificate that is specified for the OnPremLocalAgent certificate either isn’t valid or isn’t correctly configured for the tenant.

**Steps:** Follow these steps to resolve the error.

1. Run `Test-D365FOConfiguration.ps1` on all orchestrator nodes to make sure that all checks pass.
2. Verify that the certificate that is specified in the local agent configuration is correct.
   - Make sure that the thumbprint that you specify in LCS and in the ConfigTemplate.xml file has no special characters.
   - The certificate should be the same certificate that is specified in the following section in `infrastructure\ConfigTemplate.xml`.

   ```xml
   <Certificate type="Orchestrator" exportable="true" generateSelfSignedCert="true">
     <Name>OnPremLocalAgent</Name>
     <Thumbprint/></Thumbprint>
     <ProtectTo></ProtectTo>
   </Certificate>
   ```

3. Make sure that the same certificate that is specified in the local agent configuration in LCS was used to complete the steps in the "Configure LCS connectivity for the tenant" section of the appropriate setup and deployment topic for your environment:
   - Platform update 41 and later
   - Platform updates 12 through 40
4. Uninstall the local agent.
5. Specify the correct certificate in the local agent configuration, and download the configuration file again.
6. Install the local agent again by using the new configuration file.
Error

During servicing, you receive an "Unable to download asset" error, and the details state, "The credentials supplied to the package were not recognized."

Reason: The ACL wasn't correctly defined on certificates.

Steps:

Check whether ACL was removed from client certificate on orchestrator machines. Run the \Test-D365FOConfiguration.ps1 script on orchestrator machines, and verify the ACL.

To resolve the error, run the \Set-CertificateAcls.ps1 script to reset the ACLs.

Error

Access to the path 'C:\\\agent\assets\StandAloneSetup-76308-1.zip' is denied.

Reason: The file share that is specified in the local agent configuration isn't valid.

Steps: Follow these steps to resolve the error.

1. Verify that the specified share exists.
2. Verify that the local agent user has full permission on the share. The local agent user is the Domain Name System (DNS) name that is specified in the following section in ConfigTemplate.xml.

```xml
<ADServiceAccount type="gMSA" name="svc-LocalAgent$" refName="gmsaLocalAgent">
  <DNSHostName>svc-LocalAgent.d365fo.onprem.contoso.com</DNSHostName>
</ADServiceAccount>
```

3. Make sure that the "Set up file storage" section of the appropriate setup and deployment topic for your environment is completed:
   - Platform update 41 and later
   - Platform updates 12 through 40
4. Uninstall the local agent.
5. Specify the correct file share in the local agent configuration, and download the configuration file again.
6. Install the local agent again by using the new configuration file.

Error

When you do a servicing operation, you receive the following error:

Unable to get extract setup folder for command

Reason: The file share has been removed or changed.

Steps: To see what the file share is set to, open Microsoft SQL Server Management Studio, and run the following query on the orchestrator database:

```
select * from OrchestratorCommandArtifact where CommandId = 'xxx'
```
Error: Login failed for user 'D365\svc-LocalAgent$'. Reason: Could not find a login matching the name provided. [CLIENT: 10.0.2.23]

**Reason:** The local agent user can't connect to the orchestrator database. This issue can occur because users have been deleted and then re-created in Active Directory Domain Services (AD DS). Therefore, the security identifier (SID) of the user has changed, and any access that was given to the user for the SQL Server instance or the database no longer works.

**Steps:** Follow these steps to resolve the error.

1. Run the following script on the SQL Server instance.

   ```bash
   .\Initialize-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName Orchestrator
   ```

   This script creates an empty orchestrator database, if an empty database doesn't already exist. It then adds the local agent user to the database and gives it db_owner permission.

   After the correct permissions are provided, the application should automatically go to a healthy state.

2. If any settings, such as the fully qualified domain name (FQDN) of the SQL Server instance, the database name, or the local agent user, were provided incorrectly in LCS, change the settings, and then reinstall the local agent.

   If the preceding steps don't resolve the error, manually remove the local agent user from the SQL Server instance and the database, and then rerun the Initialize-Database script.

   If you re-create a user in AD DS, remember that the SID will change. In this case, remove the previous SID for the user, and add a new SID.

---

**Error**

**Steps:**

- Verify that you have access to the SQL Server listener.
- If you’re doing testing, you can start over and use an empty orchestrator database.

---

**Issue**

When you performing the Configure the databases procedure, if the SQL Server instance is a named instance, use the `-DatabaseServer [FQDN/Instancename]` parameter.

---

**Issue**

The local agent user can't connect to the SQL Server instance or the database.

**Steps:** Follow these steps to resolve the error.

1. Delete the svc-LocalAgent user from the SQL Server primary node databases, and then remove the login from both servers.

2. Run the following scripts.

   ```bash
   .\Initialize-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName Orchestrator
   .\Configure-Database.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ComponentName Orchestrator
   ```
**Infrastructure scripts errors**

**Issue**

*Error*: When you run Test-D365FOConfiguration.ps1 or Test-D365FOConfiguration-AllVMs.ps1, you receive the message:

```
"Get-LocalGroupMember : Failed to compare two elements in the array.
At C:\Infrastructure\Scripts\Test-D365FOConfiguration.ps1:79 char:9
  + Get-LocalGroupMember -Group 'Administrators' | `
  + CategoryInfo          : NotSpecified: (:) [Get-LocalGroupMember], InvalidOperationException
  + FullyQualifiedErrorId : An unspecified error
  occurred.,Microsoft.PowerShell.Commands.GetLocalGroupMemberCommand"
```

*Reason*: There is a bug in the PowerShell commandlet, Get-LocalGroupMember, which causes it to fail when there are entries that not valid.

*Steps*: On the machine where the script is failing, open **local users and groups**. Go to the administrators group and remove any entries that have an entry like the one highlighted in the following image.

---

**IMPORTANT**

These scripts don't work when an **always-on** setup is used. The database must first be created in the primary node and then replicated.

---

**Error**

*Error*:

```
RunAsync failed due to an unhandled exception causing the host process to crash:
System.Net.Http.HttpRequestException: An error occurred while sending the request. --->
System.Net.WebException: The remote name could not be resolved: 'lcsapi.lcs.dynamics.com'
```

*Reason*: The local agent machines can't connect to lcsapi.lcs.dynamics.com. Review the AX-BridgeService event log for "The remote name could not be resolved: 'lcsapi.lcs.dynamics.com'."

*Steps*: Follow these steps to resolve the error.

1. Run `psping lcsapi.lcs.dynamics.com:80`.
2. If you don't receive a response from the preceding command, contact the IT department at your organization. Either the firewall is blocking access to lcsapi, or proxy issues are occurring.

```
lcsapi.lcs.dynamics.com:443
login.windows.net:443
uswelcs1lcm.queue.core.windows.net:443
www.office.com:443
login.microsoftonline.com:443
dc.services.visualstudio.com:443
uswelcs1lcm.blob.core.windows.net:443
uswedpl1catalog.blob.core.windows.net:443
```

---

**Error**

*Error*:

```
An error occurred while sending the request. --->
System.Net.WebException: The remote name could not be resolved: 'lcsapi.lcs.dynamics.com'
```

*Reason*: The local agent machines can't connect to lcsapi.lcs.dynamics.com. Review the AX-BridgeService event log for "The remote name could not be resolved: 'lcsapi.lcs.dynamics.com'."

*Steps*: Follow these steps to resolve the error.

1. Run `psping lcsapi.lcs.dynamics.com:80`.
2. If you don't receive a response from the preceding command, contact the IT department at your organization. Either the firewall is blocking access to lcsapi, or proxy issues are occurring.

```
lcsapi.lcs.dynamics.com:443
login.windows.net:443
uswelcs1lcm.queue.core.windows.net:443
www.office.com:443
login.microsoftonline.com:443
dc.services.visualstudio.com:443
uswelcs1lcm.blob.core.windows.net:443
uswedpl1catalog.blob.core.windows.net:443
```

---

**Infrastructure scripts errors**

**Issue**

*Error*: When you run Test-D365FOConfiguration.ps1 or Test-D365FOConfiguration-AllVMs.ps1, you receive the message:

```
"Get-LocalGroupMember : Failed to compare two elements in the array.
At C:\Infrastructure\Scripts\Test-D365FOConfiguration.ps1:79 char:9
  + Get-LocalGroupMember -Group 'Administrators' | `
  + CategoryInfo          : NotSpecified: (:) [Get-LocalGroupMember], InvalidOperationException
  + FullyQualifiedErrorId : An unspecified error
  occurred.,Microsoft.PowerShell.Commands.GetLocalGroupMemberCommand"
```

*Reason*: There is a bug in the PowerShell commandlet, Get-LocalGroupMember, which causes it to fail when there are entries that not valid.

*Steps*: On the machine where the script is failing, open **local users and groups**. Go to the administrators group and remove any entries that have an entry like the one highlighted in the following image.
Do this on all of the machines that receive this error. After the changes are complete, try running the script again.

Restart applications (such as AOS)

In Service Fabric, expand Nodes > AOSx > fabric/AXSF > AXSF > Code Packages > Code. Select the ellipsis button (...), and then select Restart. When you're prompted, enter the code.

Upgrade Service Fabric

Service Fabric Explorer will show a message that resembles the following message:


Because the minimum requirement is one Microsoft SQL Server Reporting Services (SSRS) node and one Management Reporter node, you must pass in a parameter to skip PreUpgradeSafetyCheck.

Follow these steps to upgrade Service Fabric in Windows PowerShell.

1. Connect to the Service Fabric cluster. In the following command, replace 123 with the server/star thumbprint, and use the appropriate IP address.

   ```powershell
   Connect-ServiceFabricCluster -connectionEndpoint 10.0.0.12:19000 -X509Credential -FindType FindByThumbprint -FindValue 123 -ServerCertThumbprint 123
   ```

2. Get the latest version that was downloaded.

   ```powershell
   Get-ServiceFabricRegisteredClusterCodeVersion
   ```

NOTE

-UpgradeReplicaSetCheckTimeout is used to skip PreUpgradeSafetyCheck for SSRS and Management Reporter. For more information, see Service Fabric service upgrade not working. You might also want to use -UpgradeDomainTimeoutSec 600 -UpgradeTimeoutSec 1800. For more information, see Application upgrade parameters.


4. Get the upgrade status.

Get-ServiceFabricClusterUpgrade

For more information, see Troubleshoot application upgrades.

To learn when a new Service Fabric release comes out, see the Azure Service Fabric team blog.

If you receive a warning in Service Fabric Explorer after you upgrade, make a note of the node, and then restart by expanding Nodes > AOSx > fabric:/AXSF > AXSF > Code Packages > Code. Select the ellipsis button (...), and then select Restart.

Error: "Unable to load DLL 'FabricClient.dll"

If you receive an error that states, "Unable to load DLL 'FabricClient.dll," close and restart Windows PowerShell. If the error persists, restart the machine.

What cluster ID should be used in the agent configuration?

The cluster ID can be any globally unique identifier (GUID). This GUID is used for tracking purposes.

Encryption errors


You might receive one of these errors if the data encipherment certificate that was used to encrypt the AOS account password wasn't installed on the machine. This certificate might be in the certificates (local computer), or the provider type might be incorrect.

To resolve the error, validate the credentials.json file. Verify that the text is correctly decrypted by entering the following command (on AOS1).

Invoke-ServiceFabricDecryptText -CipherText 'longstring' -StoreLocation LocalMachine | Set-Clipboard

This error can also occur if the "" parameter isn't defined in the ApplicationManifest file. To determine whether this parameter is defined, in Event Viewer, go to Custom Views > Administrative Events, and verify the following information:

- The encrypt credentials for the credentials.json file have the correct layout/structure. For more information, see the "Encrypt credentials" section of the appropriate setup and deployment topic for your environment:
Properties to create a DataEncryption certificate

Use the following properties to create the DataEncryption certificate:

- **Is self-signed certificate** – Enable this parameter only when you're using self-signed certificates.
- **Certificate purposes** – Enable all purposes for this certificate.
- **Signature algorithm** – Specify `sha256RSA`.
- **Signature hash algorithm** – Specify `sha256`.
- **Issuer** – Specify `CN = DataEncryptionCertificate`.
- **Public Key** – Specify RSA (2048 bits).
- **Thumbprint algorithm** – Specify `sha1`.

**WARNING**

Don't use self-signed certificates in production environments. Instead, use certificates that are issued by certificate authorities.

The certificate and private key that should be used for decryption can't be found (0x8009200C)

If you're missing a certificate and ACL, or if you have the wrong thumbprint entry, check for special characters, and look for thumbprints in `C:\ProgramData\SF\<AOSMachineName>\Fabric\work\Applications\AXBootstrapperAppType_App<N>\log\ConfigureCertificates-<timestamp>.txt`.

You can also validate the encrypted text by using the following command.

```
Invoke-ServiceFabricDecryptText -CipherText 'longstring' -StoreLocation LocalMachine | Set-Clipboard
```

If you receive the message, "Cannot find the certificate and private key to use for decryption," verify the `axdataenciphermentcert` and `svc-AXSF$ AXServiceUser` ACLs.

If the credentials.json file has changed, the action you should take depends on the status of the environment in LCS.

- If your environment appears to be deployed in LCS, do the following:
  1. Go to your environment page and select **Maintain**.
  2. Select **Update settings**.
  3. Do not change any settings. Select **Prepare**.
  4. After a few minutes your environment will be prepared and you can select **Deploy**.
- If your environment is in a failed state in LCS, do the following:
  1. Select **Retry**. The new Credentials.json file will be used during the retry operation.

If none of the preceding solutions work, follow these steps.
1. Verify that the domain name and Active Directory account names that are specified in the ConfigTemplate.xml file are correct.

2. Verify that the thumbprints that are specified in the ConfigTemplate.xml file are correct if the certificate wasn't generated by using the scripts that are provided.

3. Verify that the certificate thumbprints that are specified in LCS are correct, and that they match the thumbprints that are specified in ConfigTemplate.xml. Make sure that there are no special characters. You can run \Get-DeploymentSettings.ps1 to obtain the thumbprints in an easy-to-copy manner.

4. If the certificates aren't self-generated, make sure that the provider names match for the following certificate types:
   - **ServiceFabricEncryption type**: Microsoft Enhanced Cryptographic Provider v1.0
   - **All other certificate types**: Microsoft Enhanced RSA and AES Cryptographic Provider

5. Verify that the Set-CertificateAcls.ps1 and Test-D365FOConfiguration.ps1 scripts were successfully run on all Service Fabric machines.

6. Verify that the credentials.json file exists, and that the entries are decrypted to correct values.

   On one of the AOS machines, run the following command to verify that the data encryption certificate is correct.

   ```bash
   Invoke-ServiceFabricDecryptText '<encrypted string>' -StoreLocation LocalMachine
   ```

7. If any of the certificates must be changed, or if the configuration was incorrect, follow these steps:
   a. Edit the ConfigTemplate.xml file so that it has the correct values.
   b. Run all the setup scripts and the Test-D365FOConfiguration script.

8. In LCS, reconfigure the environment.

### Gateway fails to deploy

**Issue:** You receive the following error in the event viewer logs.

```
Message Module aos failed. Detail: System.InvalidOperationException: Gateway app and Bootstrapper app are not healthy at AOSSetupHybridCloud.Program.Main(String[] args)
at System.AppDomain._nExecuteAssembly(RuntimeAssembly assembly, String[] args)
at System.AppDomain.ExecuteAssembly(String assemblyFile, String[] args)
at System.AppDomain.ExecuteAssembly(String assemblyFile, String[] args)
at SetupCore.SetupManager.LaunchProcessInAppDomain(String startupExe, String workingDir, String currentFolder, String[] moduleArgs)
at SetupCore.SetupManager.<>c__DisplayClass12_1.<InvokeModules>b__6()
```

You also receive the following error message in the SFExplorer for the Gateway application.
'System.RA' reported Warning for property 'ReplicaOpenStatus'.
Replica had multiple failures during open on AOS_13. API call: IStatelessServiceInstance.Open(); Error = System.InvalidOperationException (-2146233079)
Category does not exist.
   at System.Diagnostics.PerformanceCounterLib.CounterExists(String machine, String category, String counter)
   at System.Diagnostics.PerformanceCounter.InitializeImpl()
   at System.Diagnostics.PerformanceCounter..ctor(String categoryName, String counterName, String instanceName, Boolean readOnly)
   at System.Diagnostics.PerformanceCounter..ctor(String categoryName, String counterName, String instanceName)
   at Microsoft.Dynamics.LBD.Gateways.ClusterGateway.Helpers.CpuPerfCounter..ctor()
   at Microsoft.Dynamics.LBD.Gateways.ClusterGateway.GzipContentDelegatingHandler..ctor()

**Reason:** The pointers to the performance counter that the gateway needs may be corrupt.

**Resolution:** Run `lodctr /R` in a Command Prompt window that you open as administrator in all AOS nodes where the gateway is unhealthy. If you receive an error message that states that the performance counters can't be rebuilt, try to run the command again.

**Management Reporter**

Additional logging can be done by registering providers. To do this, download the **LBDMRDeployerTroubleshooter** asset from the LCS Shared Asset Library. You can find the asset in the model asset type. Copy the zip file to the primary orchestrator machine, extract it, and then run the following commands. To determine which machine is the primary instance, in Service Fabric Explorer, expand ` Cluster > Applications > LocalAgentType > fabric:/LocalAgent/OrchestrationService > (GUID).`

**NOTE**

If results in Event Viewer don't appear correct (for example, if words are truncated), get the latest manifest and .dll files. To get the latest manifest and .dll files, go to the WP folder in the agent file share. This share was created in the “Set up file storage” section of the appropriate setup and deployment topic for your environment:

- Platform update 41 and later
- Platform updates 12 through 40

**Example:** `\Agent Share\wp\[Deployment name]\StandaloneSetup-...\Apps\ETWManifests`

If you must unregister providers, use the following command.

```
.\RegisterETW.ps1 -ManifestsAndDll @"C:\Files\ETWManifest\Microsoft.Dynamics.Reporting.Instrumentation.man" = "C:\Files\ETWManifest\Microsoft.Dynamics.Reporting.Instrumentation.dll"
```

After providers are registered, additional details about the new deployment are logged in Event Viewer, at **Applications and Services Logs > Microsoft > Dynamics.** The following folders will be shown:

- MR-Logger
- MR-Sql
To see the new folders, you must close and reopen Event Viewer. To see additional details, you must deploy an environment again.

**Could not load file or assembly EntityFramework**

**Issue**: You are running Local Agent version 2.3.1 or later and you received the following stacktrace in the event logs while deploying a package that contains Platform update 29 or earlier:

```
System.Reflection.TargetInvocationException: Exception has been thrown by the target of an invocation. --->
System.IO.FileNotFoundException: Could not load file or assembly 'EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089' or one of its dependencies. The system cannot find the file specified. at Microsoft.Dynamics.Integration.Service.Utility.AdapterProvider.RefreshAdapters()
--- End of inner exception stack trace ---
```

**Resolution**: Use TSG_UpdateFRDeployerConfig.ps1. For more information, see TSG_UpdateFRDeployerConfig.ps1.

**Unable to deploy Financial Reporting Service**

**Issue**: You are unable to finish deployment of Platform update 26 and later for Financial Reporting because the following error is in the application log for Service Fabric.

```
Description: The process was terminated due to an unhandled exception.
```

**Reason**: The Microsoft Visual C++ Redistributable Package for Visual Studio 2013 was not correctly installed or is corrupt in some or all of the MR nodes.

**Steps**: Re-run the installation of the Microsoft Visual C++ Redistributable Package for Visual Studio 2013.

**An error occurs while AddAXDatabaseChangeTracking is running**

If you receive an error while you run AddAXDatabaseChangeTracking at Microsoft.Dynamics.Performance.Deployment.FinancialReportingDeployerUtility.InvokeCmdletAndValidateSuccess(DeploymentCmdlet cmdlet), verify that the full path is correct. An example of a full path is ax.d365ffo.onprem.contoso.com.

The error might also occur because of an issue with the star certificate. For example, the remote certificate CN=*.d365ffo.onprem.contoso.com has a name that isn't valid or that doesn't match the host, ax.d365ffo.onprem.contoso.com.

**Run the initialize database script, and validate that databases have correct users**

If you receive only the AddAXDatabaseChangeTracking event, try to reach the MetadataService service for Finance + Operations by going to https://ax.d365ffo.contoso.com/namespaces/AXSF/services/MetadataService.

Next, check the certificates of the service in the wif.config file. To find the file, sign in to one of the AOS machines, and then, in Task Manager, find AxService.exe. Right-click, and select Open file location. In the wif.config file,
you should see three thumbprints. Note the following requirements for these thumbprints:

- They must be different.
- They must be in this order:
  1. Financialreporting thumbprint
  2. ReportingService thumbprint
  3. SessionAuthentication thumbprint

If the thumbprints don’t meet both these requirements, you must redeploy from LCS by using correct thumbprints.

**The remote name can't be resolved**

**Error:**

The remote name could not be resolved: ‘x.d365fo.onprem.contoso.com’ / There was no endpoint listening at https://x.d365fo.onprem.contoso.com/namespaces/AXSF/services/MetadataService that could accept the message.

**Reason:** This issue is often caused by an incorrect address or SOAP action.

**Steps:** Verify that the address can be reached, by manually opening the URL. For more details, see the “InnerException” text in the Event Viewer, if it’s present.

**Error on ImportDefaultReports**

If Management Reporter reports are checked out during deployment, the deployment will fail. To see whether reports are checked out, run the following select statements on the FinancialReporting database.

```sql
select checkedoutto, * from Reporting.ControlReport where checkedoutto is not null
select checkedoutto, * from Reporting.ControlRowMaster where checkedoutto is not null
select checkedoutto, * from Reporting.ControlColumnMaster where checkedoutto is not null
```

To learn which user has objects checked out, you can run the following select statement.

```sql
select * from Reporting.SecurityUser where UserID = ''
```

To resolve this issue manually, update the following tables, and set checkedoutto to null by using the following commands.

```sql
update Reporting.ControlReport set checkedoutto = null where checkedoutto is not null
update Reporting.ControlRowMaster set checkedoutto = null where checkedoutto is not null
update Reporting.ControlColumnMaster set checkedoutto = null where checkedoutto is not null
```

**axdbadmin can't connect to the database server SQL-LS.contoso.com**

**Reason:** The user doesn't have permission to connect to the AXDB database.

**Steps:**

1. Remove the axdbadmin user from the database, if it already exists.
2. In the ConfigTemplate.xml file, specify the user name that must be added to the AXDB database.
3. Run the initialize database script again to add the axdbadmin user.

Unable to resolve the xPath value

In the expected behavior, the following xPath value can't be resolved:

`[TopologyInstance/CustomizationGroup[@name='ServiceConfiguration']/Group[@name='AOSServicePrincipalUser']/Customizations/Customization[@fieldName='PrincipalUserAccountPassword']/@selectedValue`

Therefore, the fact that the xPath value can't be resolved isn't an issue. The xPath value looks for AOS runtime user information. However, because of integrated security, that information isn't required. The fact that the xPath value can't be resolved is communicated in case the failure must be investigated for another reason.

AD FS

The sign-in page doesn't redirect you

The sign-in page might not redirect you but continues to prompt for credentials. Alternatively, you might be redirected but receive the following message:

```
An error occurred. Contact your administrator for more information.
```

In these cases, you can follow these steps to resolve the issue:

- Add the AD FS link to the list of trusted sites.
- Add the Dynamics 365 link to the list of trusted sites.
- Add a trailing slash (/), and see whether the behavior changes.

Verify the AD FS Manager by going to ADFS > Application groups. Double-click Microsoft Dynamics 365 for Operations on-premises. Then, under Native application, double-click Microsoft Dynamics 365 for Operations on-premises - Native application.

Note the Redirect URI value. It should match the DNS forward lookup zone for Finance + Operations.

Error: "Could not establish trust relationship for the SSL/TLS secure channel"

If you receive an error that states, "Could not establish trust relationship for the SSL/TLS secure channel," follow these steps.

1. In Service Fabric, go to Cluster > Applications > AXSFTYPE > fabric/AXSF, and then, on the Details tab, scroll down and note the URLs for Aad_AADMetadataLocationFormat and Aad_FederationMetadataLocation.

2. Browse to those URLs from AOS.

3. On the AOS machine, in Event Viewer, go to Applications and Services Logs > Microsoft > Dynamics > AX-SystemRuntime for details.

4. Verify that the AD FS certificate is trusted:

   a. Verify the AD FS certificate. On the AD FS machine, in Server Manager, go to Tools > AD FS Management.

   b. Expand AD FS > Service > Certificates, and make a note of the certificates. For example, one certificate might be dc1.contoso.com.
c. On the AOS machine, in the Microsoft Management Console Certificates snap-in, go to Certificates (Local Computer) > Trusted Root Certification Authorities > Certificates, and verify that the AD FS certificate is listed.

If you receive a message that states that the site isn’t secure, you haven’t added your Secure Sockets Layer (SSL) certificate for AD FS to the Trusted Root Certification Authorities store.

**You can’t connect to the remote server in some locations**

You might not be able to connect to the remote server at the following places:

- System.Xml.XmlDownloadManager.GetNonFileStream(Uri uri, ICredentials credentials, IWebProxy proxy, RequestCachePolicy cachePolicy)
- System.Xml.XmlUrlResolver.GetEntity(Uri absoluteUri, String role, Type ofObjectToReturn)

In this case, go to the C:\ProgramData\SF\AOS_1\Fabric\work\Applications\AXSFTApp35\log folder where you receive the error, and note the out file. The out file contains the following information:

```plaintext
System.Net.WebException: Unable to connect to the remote server --->
System.Net.Sockets.SocketException: A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond x.x.x.x:443
```

You can also use Psing to try to reach the remote server. For information about Psing, see Psing.

**Redirect sign-in questions and issues**

If you’re having issues when you sign-in, in Service Fabric Explorer, verify that the Provisioning_AdminPrincipalName and Provisioning_AdminIdentityProvider values are valid. Here is an example:

- **Provisioning_AdminPrincipalName**: AXServiceUser@contoso.com
- **Provisioning_AdminIdentityProvider**: https://DC1.contoso.com/adfs

If the values aren't valid, you won’t be able to proceed, and you must redeploy from LCS.

If you used Reset-DatabaseUsers.ps1, you must restart the Dynamics Service before your changes take effect. If you still have sign-in issues, in the USERINFO table, note the NETWORKDOMAIN and NETWORKALIAS values. Here is an example:

- **NETWORKDOMAIN**: https://DC1.contoso.com/adfs
- **NETWORKALIAS**: AXServiceUser@contoso.com
- **IDENTITYPROVIDER**: This should match the NETWORKDOMAIN value.

On the AD FS machine, in Server Manager, go to Tools > AD FS Management > Service. Right-click Service and Edit Federation Service Properties. The Federation Service identifier value should match the USERINFO.NETWORKDOMAIN value, and it should have https in the URL (for example, https://DC1.contoso.com/adfs).

On the AD FS machine, in Event Viewer, go to Applications and Services Logs > AD FS > Admin, and make a note of any errors.

**Fiddler**

Fiddler can be used for additional debugging. For in-depth information about Fiddler, see AD FS 2.0: How to Use Fiddler Web Debugger to Analyze a WS-Federation Passive Sign-In and Cracking the AD FS Token from another AD FS Claims Provider.
The following sections provide focused debugging steps for claims that are returned to Microsoft Dynamics.

**Repo/capture**
1. Open Fiddler, go to **Tools > Options > HTTPS**, and select **Decrypt HTTPS traffic**.
2. Start to capture traffic (the shortcut key is F12). You can verify that that traffic is being captured by looking at the lower left of the tool.
3. Open an InPrivate instance of Internet Explorer or an Incognito instance of Chrome.
5. Sign in by using the USERINFO.NETWORKALIAS account and password.
6. After you're signed in, stop Fiddler from capturing traffic.

**Analyze**
In the right pane of Fiddler, notice that a horizontal divider separates the request from the response. Unlike a network trace, where you typically get one frame for a request and another frame for a response, Fiddler provides one frame that contains both the request and the response.

1. In Fiddler, in the upper-right corner, select **Inspectors > Raw**.
2. In the lower-right corner, select **Cookies**.
3. Do a search for **MSISAuth**.
4. Select the row that has a result of **200** for the AD FS host.
5. Look above the row that you just selected to find a row that has a result of **302**. Select the row.

   You should see the AD FS URL, host, user name, and password.

   **IMPORTANT**
   For privacy, you might have to scrub personally identifiable information.

6. Select the next row that has a result of **302**. The URL should be `.../namespaces/AXSF/`.
7. Find the code line that is shown on that row.
8. Copy the value of code line after the equal sign (=).
9. Go to [https://www.base64decode.org/](https://www.base64decode.org/), and paste the code that you just copied.
10. In the Source charset field, select ASCII.

11. Select Decode.

12. Copy the results, and follow these steps:
   - Make sure that the upn value matches the user name.
   - Make sure that the unique_name value is the Active Directory user that is being tested.
   - Go to Active Directory Users and Computers > domain > Users, and make sure that this user is being tested.

Sign-in issues

If you or other users experience sign-in issues, in Service Fabric Explorer, verify that the Provisioning_AdminPrincipalName and Provisioning_AdminIdentityProvider values are valid. If the values are valid, run the following command on the primary SQL Server machine.

```
Reset-DatabaseUsers.ps1
```

On each AOS machine, in Task Manager, select AXService.exe, and then select End task.

To verify that a user has been reset, run the following `select` query in the AXDB SQL database.

```
select SID, NETWORKDOMAIN, NETWORKALIAS, * from AXDB.dbo.USERINFO where id = 'admin'
```

In an Azure Active Directory (Azure AD) environment (that is, an online environment), the SID is a hash of a network alias and a network domain. In an AD DS environment (that is, an on-premises environment), the SID is a hash of a network alias and an identity provider.

In some cases, you still might not be able to sign in, and you might receive the following error:

```
You are not authorized to login with your current credentials. You will be redirected to the login page in a few seconds.
```

If this error occurs, follow these steps.

1. On the AD FS machine, go to Server Manager > Tools > AD FS Management.
2. Right-click AD FS, and then select Edit Federation Service Properties.
3. Make sure that the Federation Service Identifier value matches the Userinfo.NetworkDomain and Userinfo.IdentityProvider values.
5. Make sure that the IdTokenIssuer value matches the Federation Service Identifier value from step 3, and also the Provisioning_AdminIdentityProvider value on the fabric/AXSF Details tab at Service Fabric Explorer > Cluster > Applications > AXSFType.
6. In Service Fabric Explorer, verify that the Provisioning_AdminPrincipalName and Provisioning_AdminIdentityProvider values are valid.

If the preceding steps don't resolve the issue, see the AD FS section of this topic.

System.Data.SqlClient.SqlException (0x80131904) and
System.ComponentModel.Win32Exception (0x80004005)

You might receive one of the following errors:

System.Data.SqlClient.SqlException (0x80131904): A connection was successfully established with the server, but then an error occurred during the sign-in process. (provider: SSL Provider, error: 0 - The certificate chain was issued by an authority that is not trusted.)

System.ComponentModel.Win32Exception (0x80004005): The certificate chain was issued by an authority that is not trusted

In this case, either the certificates haven't been installed, or they haven't given access to the correct users. To resolve this error, add the public key SQL Server certificate to all the Service Fabric nodes.

Keyset doesn't exist

If you find that the keyset doesn't exist, scripts weren't run on all machines. Review and complete the "Set up VMs" section of the appropriate setup and deployment topic for your environment:

- **Platform update 41 and later**
- **Platform updates 12 through 40**

Copy the scripts in each folder to the VMs that correspond to the folder name.

Additionally, check the .csv file to verify that the correct domain is used.

Error: "RunAsync failed due to an unhandled FabricException causing replica to fault"

You might receive the following error:

RunAsync failed due to an unhandled FabricException causing replica to fault: System.Fabric.FabricException: The first Fabric upgrade must specify both the code and config versions. Requested value: 0.0.0.0:

In this case, in the ClusterConfig.json file, change **diagnosticsStore** from a network share to a local path. For example, change **\server\path** to a default value of **C:\ProgramData\SF\DiagnosticsStore**.

Service Fabric AOS node error during build: The execution time-out expired

Error:

The timeout period elapsed prior to completion of the operation or the server is not responding.
The statement has been terminated.

Only one AOS machine can run DB Sync at a time. You can safely ignore this error, because it means that one of the AOS VMs is running DB Sync. Therefore, the other VMs produce a warning that they can't run it. To verify that DB Sync is running, on the AOS VM that isn't producing warnings, in Event Viewer, go to **Applications and Services Log > Microsoft > Dynamics > AX-DatabaseSynchronize/Operational**.

Error: "RequireNonce is 'true' (default) but validationContext.Nonce is null"
You might receive the following error:

```
RequireNonce is 'true' (default) but validationContext.Nonce is null
```

This error also appears as an HTTP error 500 in Internet Explorer after you sign in to the client. The nonce that is issued can’t be validated if Internet Explorer is in Enhanced Security Configuration.

To sign in to the client, disable Enhanced Security Configuration for Internet Explorer via Server Manager.

**Error: "Invalid algorithm specified / Cryptography"**

If you receive an "Invalid algorithm specified / Cryptography" error, you must use the Microsoft Enhanced RSA and AES Cryptographic Provider. For more information, see the certificate requirements. Additionally, verify that the structure of the credentials.json file is correct.

If you must re-create the certificate by using the correct provider, follow these steps.

1. Create the certificate again by using the correct provider.
2. Change the `ConfigTemplate.xml` file.
3. Run the infrastructure scripts on all machines in the cluster, and make sure that the `Test-D365FOConfiguration.ps1` script passes.
4. Reconfigure the environment from LCS.

An "Unable to find certificate" error occurs when you run `Test-D365FOConfiguration.ps1`

If you receive an "Unable to find certificate" error when you run `Test-D365FOConfiguration.ps1`, check whether certificates or thumbprints are being combined for multiple purposes. For example, you will receive this error if the client certificate and the SessionAuthentication certificate are combined. We recommend that you not combine certificates. For more information, see the certificate requirements, and check the acl.csv file for `domain.com\user` versus `domain\user` (for example, NETBIOS structure).

**The client and server can’t communicate because they don’t have a common algorithm**

If the client and server can’t communicate because they don’t have a common algorithm, verify that the certificates that are created use the specified provider, as explained in the "Plan and acquire your certificates" section of the appropriate setup and deployment topic for your environment:

- Platform update 41 and later
- Platform updates 12 through 40

**Find a list of group managed service accounts**

To find a list of all groups and hosts, run the following command.

```
Get-ADServiceAccount -Identity svc-LocalAgent$ -Properties PrincipalsAllowedToRetrieveManagedPassword
```

**AddCertToServicePrincipal script fails on Import-Module**

**Error:**

```
AddCertToServicePrincipal script failing on Import-Module : Could not load file or assembly
```
Steps: To resolve this issue, follow these steps.

1. Run the following command in Windows PowerShell.

   ```
   Uninstall-Module -Name AzureRM
   Install-Module AzureRM
   ```

2. Close the Windows PowerShell window, and try to run the script again.

**ReportingServicesSetup.exe error**

**Error:**

- ReportingServicesSetup.exe Error: 0 : Application fabric/ReportingService is not OK after 10 minutes
- Application: ReportingServicesBootstrapper.exe
- Framework Version: v4.0.30319
- Description: The process was terminated due to an unhandled exception.

**Reason:** If you receive this error, strong name validation is enabled in the Reporting server, but it should not be enabled.

**Steps:** To resolve this issue, run the `config-PreReq` script on the Reporting server machine.

**The requested operation requires elevation**

This issue occurs because AOS users aren't in the local administrator group, and User Account Control (UAC) hasn't been disabled correctly. To resolve the issue, follow these steps.

1. Add AOS users as local admins, as described in the "Join VMs to the domain" section of the appropriate setup and deployment topic for your environment:
   - Platform update 41 and later
   - Platform updates 12 through 40
2. Run the `Config-PreReq` script on all the AOS machines.
3. Make sure that the `Test-Configuration` script passes.
4. If UAC was changed, you must restart the machine before the changes take effect.

**Files in use errors**

If these "Files in use" errors occur, set up the exclusion rules that Service Fabric advises. For information, see **Environment setup**.

**Apply deployable packages during deployment**

**Package deployment fails because of a "path too long" exception**

Because of a 260-character limit in Microsoft Windows, deployment will fail if a package has a longer name, or if the on-premises share has the full FQDN path. If the character limit is exceeded, you receive the following error:

- `System.IO.PathTooLongException`: The specified path, file name, or both are too long. The fully qualified file
name must be less than 260 characters, and the directory name must be less than 248 characters. at System.IO.PathHelper.GetFullPathName

To work around this issue, shorten the package name, and then apply the package again. Alternatively, shorten the overall length of the share path for the on-premises assets.

**Package deployment fails because of a serialization error**

During package deployment, you might receive the following error:

Serialization version mismatch detect, make sure the runtime DLLs are in sync with the deployed metadata.
Version of file 'XXX'. Version of DLL 'XXX'

In this case, the version of the environment where the package was developed might differ from the version of the environment that the package is being deployed in.

To work around this issue, keep the development or build environments on the same version as the deployed on-premises environment. You can confirm the package version by looking in the **Additional details** section in the Asset library where the package is uploaded. To fix the error, generate the package on a version that is the same as or earlier than the version that is deployed in the on-premises environment.

**Package deployment fails because of dependencies on missing modules**

If you try to apply a package that is missing dependent modules, package application will fail, and you will receive a message that resembles the following message:

Package [dynamicsax-My_commonextension.7.0.4679.35176.nupkg has missing dependencies:
[dynamicsax-demodatasuite;dynamicsax-financialreportingadaptors;dynamicsax-fleetmanagement;dynamicsax-fleetmanagementextension;dynamicsax-publicsectorformadaptor]]

Package [dynamicsax-My_commonextension.7.0.4679.35176.nupkg has missing dependencies:
[dynamicsax-demodatasuite;dynamicsax-financialreportingadaptors;dynamicsax-fleetmanagement;dynamicsax-fleetmanagementextension;dynamicsax-fleetmanagementunittests;dynamicsax-generalledgerformadaptor;dynamicsax-publicsectorformadaptor;dynamicsax-retailformadaptor]]

Package [dynamicsax-My_uiextension.7.0.4679.35176.nupkg has missing dependencies:
[dynamicsax-demodatasuite;dynamicsax-financialreportingadaptors;dynamicsax-fiscalbooksformadaptor;dynamicsax-fleetmanagement;dynamicsax-fleetmanagementextension;]

To confirm the issue and find the missing dependencies, in Event Viewer, open **Application and Services**, and then go to **Microsoft > Dynamics > AX-SetupModuleEvents** to view events that have missing modules. For example, one of the modules that is typically missing is ApplicationFoundationFormAdaptor.

To fix this issue and successfully apply the package, either add dependent modules, or remove modules that require dependent modules. To add dependent modules, you must include the dependencies when you build the package. To remove modules, you can use ModelUtil.exe to delete a module. For more information, see Export and import models.

**Package deployment works in a one-box environment but not in the sandbox environment**

A one-box environment might have all the modules installed, whereas the sandbox environment might have only the modules that are required in order to run your production environment. If the package that was built in the dev environment has a dependency on modules that are present in the one-box environment but not in the sandbox environment, the package won't work in the sandbox environment.

To resolve this issue, look at all the modules that you're dependent on, and make sure that you don't pull any farm adapter or any other module that isn't required in the production environment. The best practice is to take
An error occurs when you sign in to on-premises environments

- **Platform update 12**: Turn off the Skype integration by going to System administration > Setup > Client performance options. When you go to the app, append `?debug=true` to the URL, as shown in the following example: [https://ax.d365ffo.onprem.contoso.com/namespaces/AXSF/?debug=true](https://ax.d365ffo.onprem.contoso.com/namespaces/AXSF/?debug=true)

- **Platform update 8 and Platform update 11**: A known issue for the Skype application programming interface (API) affects the ability to sign in to on-premises environments. Microsoft is investigating a resolution for this issue. To work around this issue, you can add `?debug=true` to the end of the URL, as shown in the following example: [https://ax.d365ffo.onprem.contoso.com/namespaces/AXSF/?debug=true](https://ax.d365ffo.onprem.contoso.com/namespaces/AXSF/?debug=true)

The local agent stops working after the tenant for the project from LCS is changed

Follow these steps to configure the local agent with the updated tenant.

1. Uninstall the local agent.

   \LocalAgentCLI.exe Cleanup <path of localagent-config.json>

2. Follow the steps in the "Configure LCS connectivity for the tenant" section of the appropriate setup and deployment topic for your environment:

   - **Platform update 41 and later**
   - **Platform updates 12 through 40**

3. Create a new LCS connector in the new tenant.

4. Download the local-agent.config file.

5. Install the local agent.

   \LocalAgentCLI.exe Install <path of localagent-config.json>

Additional deployments (for example, two sandbox deployments, or a sandbox and production deployment)

You will receive the following error when you deploy an additional environment:


You can skip or modify the following sections in the deployment instructions.

**Plan and acquire your certificates (as documented for Platform update 42 and later or Platform updates 12 through 40)**

- You must use the same on-premises local agent certificate.
- You can use same star certificates (AOS SSL and Service Fabric).
- The remaining certificates should probably differ from the certificates for the existing environment.

**Download setup scripts from LCS (as documented for Platform update 42 and later or Platform updates 12 through 40)**
- The scripts that are downloaded should be copied into a new folder.

**Set up a standalone Service Fabric cluster (as documented for Platform update 42 and later or Platform updates 12 through 40)**
- The scripts that are downloaded should be copied into a new folder.

**Configure LCS connectivity for the tenant (as documented for Platform update 42 and later or Platform updates 12 through 40)**
- You must complete this task only one time for the tenant.

**Configure AD FS (as documented for Platform update 42 and later or Platform updates 12 through 40)**
- Configure AD FS according to the **Reuse the same AD FS instance for multiple environments** guide.

**Redeploy SSRS reports**

**Version 10.0.13 or later**
Run the following command against your business data database (AXDB):

```sql
UPDATE SF.synclog SET STATE=5, SyncStepName = 'ReportSyncstarted' WHERE CODEPACKAGEVERSION in (SELECT TOP(1) CODEPACKAGEVERSION FROM SF.SYNCLOG ORDER BY CREATIONDATE DESC)
```

**Version 10.0.12 or earlier**
Run the following command against your business data database (AXDB):

```sql
DELETE FROM SF.synclog WHERE CODEPACKAGEVERSION in (SELECT TOP(1) CODEPACKAGEVERSION FROM SF.SYNCLOG ORDER BY CODEPACKAGEVERSION DESC)
```

**NOTE**
If you are using version 10.0.12 or earlier, a full database synchronization will be executed.

After running the command, restart one of your AOS nodes through Service Fabric Explorer or restart the VM that the node is running on.

**Add axdbadmin to tempdb after a SQL Server restart via a stored procedure**
When SQL Server is restarted, the tempdb database is re-created. Therefore, there will be missing permissions.
Run the following script to create a stored procedure on the master database.

```sql
\-----
USE [master]
GO
CREATE procedure [dbo].[CREATETEMPDBPERMISSIONS] as begin
exec ('USE tempdb; declare @dbaccesscount int; select @dbaccesscount = COUNT(*) from master..syslogins where name = ''axdbadmin''; if (@dbaccesscount <> 0)
exec sp_grantdbaccess ''axdbadmin''; ALTER USER [axdbadmin] WITH DEFAULT_SCHEMA=dbo; EXEC sp_addrolemember N''db_datareader'', N''axdbadmin''; EXEC sp_addrolemember N''db_datawriter'', N''axdbadmin''; EXEC sp_addrolemember N''db_ddladmin'', N''axdbadmin''; exec sp_grantdbaccess ''contoso\svc-AXSF$''; ALTER USER [contoso\svc-AXSF$] WITH DEFAULT_SCHEMA=dbo; EXEC sp_addrolemember N''db_datareader'', N''contoso\svc-AXSF$''; EXEC sp_addrolemember N''db_datawriter'', N''contoso\svc-AXSF$''; EXEC sp_addrolemember N''db_ddladmin'', N''contoso\svc-AXSF$'';') end
GO
EXEC sp_procoption N''[dbo].[CREATETEMPDBPERMISSIONS]'', 'startup', '1'
\-----
```
Error: "Updates to existing credential with KeyId '<key>' is not allowed"

You might receive the following error:

| Updates to existing credential with KeyId '<key>' is not allowed |

The steps to resolve this issue depend on whether you have only an on-premises project, or whether you have both an online project and an on-premises project.

**If have only an on-premises project**

If have only an on-premises project, you can't update the existing credential with KeyId '<key>'.

| New-AzureRmADSpCredential : Update to existing credential with KeyId '<key>' is not allowed. |
| At C:\InfrastructureScripts\Add-CertToServicePrincipal.ps1:62 char:1 |
| New-AzureRmADSpCredential -ObjectId $servicePrincipal.Id -CertValue $ ... |
| CategoryInfo : InvalidOperation: () [New-AzureRmADSpCredential], Exception |

Run the following PowerShell command to resolve the issue.

| Remove-AzureRmADSpCredential -ServicePrincipalName "00000015-0000-0000-c000-000000000000" -KeyId <key> |

**If you have both an online project and an on-premises project**

If you have both an online project and an on-premises project, follow these steps.

1. Verify that the Microsoft .NET Framework version 4.7.2 is installed.

2. Run the following Windows PowerShell script to install the Azure PowerShell module.

   | Install-Module -Name Az |

3. Run the following Windows PowerShell script to upload the new certificate.
Import-Module -Name Az.Accounts
Import-Module -Name Az.Resources
Connect-AzAccount

$servicePrincipalName = "00000015-0000-0000-c000-000000000000";
$certificateThumbprint = <Thumbprint of Agent Certificate>
$cert = Get-ChildItem -path Cert:\CurrentUser\my | Where-Object { $_.Thumbprint -eq $certificateThumbprint }
if (!$cert)
{
    $cert = Get-ChildItem -path Cert:\LocalMachine\my | Where-Object { $_.Thumbprint -eq $certificateThumbprint }
    if (!$cert)
    {
        throw "Unable to find the certificate in the Local machine or Current User store"
    }
}
$keyValue = [System.Convert]::ToBase64String($cert.GetRawCertData())
$servicePrincipal = Get-AzADServicePrincipal -ServicePrincipalName $servicePrincipalName
if (!$servicePrincipal)
{
    throw "Unable to find the service principal"
}
New-AzADSpCredential -ObjectId $servicePrincipal.Id -CertValue $keyValue -EndDate $cert.NotAfter -StartDate $cert.NotBefore
Get-AzADSpCredential -ObjectId $servicePrincipal.Id

4. Run the following command to remove the duplicate certificate, if more than one certificate exists.

Remove-AzADSpCredential -ServicePrincipalName "00000015-0000-0000-c000-000000000000" -KeyId <key>

ODBC driver 17 is required for platform updates

The latest platform binary update uses Open Database Connectivity (ODBC) driver 17. This upgrade resolves stability issues that are linked to older ODBC drivers. The Setup perquisites documentation has been updated to reflect the change in which ODBC driver 17 must be installed on each AOS server. If you don't install ODBC driver 17, you will receive DB Sync errors during servicing of the environment.

Here are some examples of errors:

- In Service Fabric:
  Unhealthy event: SourceId='System.RA', Property='ReplicaOpenStatus', HealthState='Warning', ConsiderWarningAsError=false.
  Replica had multiple failures during open on AOS3. API call: IStatelessServiceInstance.Open(); Error = System.Exception (-2146233088)
  DB sync failed.

- On AOS machines:
  - Event Viewer > Custom Views > Administrative Events:

    Description: The process was terminated due to an unhandled exception. Exception Info:
**A "No subscription found in the context" error occurs when you run Add-CertToServicePrincipal**

Recent versions of Windows PowerShell might cause a "No subscription found in the context" error. To resolve this issue, install and load an older version of Windows PowerShell, such as version 5.7.0.

```bash
# Install version 5.7.0 of Azure PowerShell
Install-Module -Name AzureRM -RequiredVersion 5.7.0

# Load version 5.7.0 of Azure PowerShell
Import-Module -Name AzureRM -RequiredVersion 5.7.0
```

**Service Fabric Explorer warnings occur after you restart a machine**

Error:

```
Error event: SourceId='MonitoringAgentService', Property='ServiceState'.
System.Management.Automation.RuntimeException: Error: The GUID passed was not recognized as valid by a WMI data provider. (Exception from HRESULT: 0x80071068). Stack trace:
```

Steps: To resolve this issue, restart the application package that generated the warning message. For more information, see Restart applications (such as AOS).

**The internal time zone version number that is stored in the database is higher than the version that is supported by the kernel (13/12)**

This database synchronization error can cause an old platform build (Platform update 12) to be deployed on top of a database that had a newer build (Platform update 15).

To resolve this issue, note the `SYSTIMEZONESVERSION` value.

```sql
select * from SQLSYSTEMVARIABLES where parm = 'SYSTIMEZONESVERSION'
```

Update the value to the version that was returned in the error message.

```sql
update SQLSYSTEMVARIABLES set VALUE = 12 where parm = 'SYSTIMEZONESVERSION'
```
Printing randomly stops

Make sure that all network printers that have been installed on the AOS server are running as the Windows service account that the AXIService.EXE process is running as.

For more information about how to configure network printers in on-premises environments, see Install network printer devices in on-premises environments.

Ax-DatabaseSynchronize isn't populated with events

In Platform update 20 and later, there is database synchronization log issue where the synchronization logs aren't written under Ax-DatabaseSynchronize in Event Viewer.

To resolve this issue, go to `<SF-dir>\AOS_<x>\Fabric\work\Applications\AXSFType_App<x>\log`. For example, go to C:\ProgramData\SF\AOS_11\Fabric\work\Applications\AXSFType_App183\log. Here, you can see the output from DatabaseSynchronize in the Code_AXSF_M_<X>.out files. Troubleshoot any issues that pertain to this component.

You can't access Finance + Operations: "AADSTS50058: A silent sign-in request was sent but no user is signed in"

After a user enters credentials to sign in to Finance + Operations, the browser briefly shows the application layout. However, it then tries to redirect outside Finance + Operations, but fails with the following error:

AADSTS50058: A silent sign-in request was sent but no user is signed in.

The cookies that represent the user's session weren't sent in the request to Azure AD. This issue can occur if the user is using Internet Explorer or Microsoft Edge, and if the web app that sends the silent sign-in request is in a different IE security zone than the Azure AD endpoint (login.microsoftonline.com).

This issue occurs because there was a change in the Skype Presence API, and on-premises environments connect to this API by default.

To resolve the issue, run the following SQL Server query.

```
update [AXDB].[dbo].[SYSCLIENTPERF] set SkypeEnabled = 0
```

Alternatively, turn off the Skype presence enabled option on the Client performance options page (System administration > Setup > Client performance options). To use this approach, you must be able to sign in to Finance + Operations. Therefore, you must first block redirection in the browser. After you disable the Skype presence, you can unblock redirection again.

The Google Chrome browser blocks redirection by default.

Error: "There was an error during CodePackage activation. Service host failed to activate. Error:0x8007052e"

You might receive the following error during a new installation:

```
Error event: SourceId='System.Hosting', Property='CodePackageActivation:Code:EntryPoint'. There was an error during CodePackage activation. Service host failed to activate. Error:0x8007052e
```

This error will cause the AXSF service to fail with the same error.
To resolve this issue, follow these steps.

1. In the **agent share path**, find the `netstandard.dll` file. For example, this file might be at `\wp\<name>\StandaloneSetup-<ver>\Apps\AOS\AXServiceApp\AXSF\Code\bin\netstandard.dll`.

2. On each AOS server, open a Command Prompt window as an administrator, and run the following command.

   ```
   "C:\Program Files (x86)\Microsoft SDKs\Windows\v8.1A\bin\NETFX 4.5.1 Tools\gacutil.exe" -i <path from step 1.>\\netstandard.dll /f
   ```

3. **Delete `AXBootstrapperApp`** from Service Fabric.
   - a. Delete the `fabric:/Bootstrapper/AXBootstrapper` service.
   - b. Delete the `fabric:/Bootstrapper` application.
   - c. Unprovision the `AXBootstrapperAppType` type.

4. **Redeploy** the environment from LCS.

**SQL Server 2016 service pack 2 is recommended for Reporting Services instances**

When you go through LCS servicing operations, you might receive the following error:

```null
The process cannot access the file 'C:\Program Files\Microsoft SQL Server\MSRS13.MSSQLSERVER\Reporting Services\ReportServer\bin\Microsoft.Dynamics.AX.Framework.Services.Platform.Client.dll' because it is being used by another process.
```

This issue occurs because Reporting Services has a lock on a Microsoft Dynamics .dll file. We currently recommend that you have SQL Server 2016 service pack 2 installed on Reporting Services instances.

**NOTE**

You must have service pack 2 installed, and no additional cumulative updates or hotfixes must be installed.

**SysClassRunner doesn’t run successfully**

**Issue:** When you try to run SysClassRunner on Platform update 29 through Platform update 31, you get the following exception:

```null
Microsoft.Dynamics.Ax.Xpp.ClrErrorException: TypeInitializationException --->
System.TypeInitializationException: The type initializer for 'Microsoft.Dynamics.Ax.Metadata.XppCompiler.CompilerTracer' throw an exception. --->
System.TypeInitializationException: The type initializer for 'Microsoft.Dynamics.Ax.DesignTime.Telemetry.OneDS' throw an exception. --->
System.IO.FileLoadException: Could not load file or assembly 'Microsoft.Diagnostics.Tracing.TraceEvent, Version=2.0.43.0, Culture=neutral, PublicKeyToken=b03f5f7f11d9c7c6' or one of its dependencies. The located assembly's manifest definition does not match the assembly reference. (Exception from HRESULT: 0x80131040) at Microsoft.Dynamics.Ax.DesignTime.Telemetry.OneDS.cctor()
--- End of inner exception stack trace ---
```

**Reason:** There is a .dll mismatch between the runtime and the application.

**Resolution:** Use `TSG_SysClassRunner.ps1`. For more information, see `TSG_SysClassRunner.ps1`. 

DBSync fails with PEAP APP version 10.0.9 Platform update 33

**Issue:** During deployment of the APP 10.0.9 PU33 PEAP-package, the deployment fails with the AXSF applications staying in "InBuild" status in Service Fabric explorer. When reviewing the logs on the AXSF nodes's work directories, the following DBSync error can be found.

Error message from DBSync:

```csharp
Microsoft.Dynamics.AX.Deployment.Setup.exe -bindir "C:\ProgramData\SF\LBDEN08FSIAOS03\Fabric\work\Applications\AXSFType_App398\AXSF.Code.1.0.20200123151456\Packages" -metadatadir "C:\ProgramData\SF\LBDEN08FSIAOS03\Fabric\work\Applications\AXSFType_App398\AXSF.Code.1.0.20200123151456\Packages" -sqluser "" -sqlserver "" -sqldatabase "" -setupmode servicesync -syncmode fullall -onprem
Stack trace: Invalid attempt to call running in CIL on the client.
  at Microsoft.Dynamics.Ax.MSIL.Interop.throwException(Int32 ExceptionValue, interpret* ip)
  at Microsoft.Dynamics.Ax.MSIL.Interop.ThrowCQLError(IL_CQL_ERR cqlErr, String p1)
  at Microsoft.Dynamics.AX.Kernel.ApplicationId.LogOrRethrow(Exception exception)
  at Microsoft.Dynamics.AX.Kernel.ApplicationId.LogOrRethrowFormattedMessage(Exception exception, String typeName, String elementName)
  at Microsoft.Dynamics.AX.Kernel.ApplicationId.LogOrRethrowFormattedMessage(Exception exception, String typeName, Int32 typeId)
  at Microsoft.Dynamics.AX.Kernel.ApplicationId.ApplicationIdBridge.LoadTableById(ApplicationIdBridge* , Int32 id, ObjectIdDelegate* cb)
  at cqlClass.callEx(cqlClass* , Char* , interpret* )
  at Microsoft.Dynamics.Ax.MSIL.cqlClassIL.Call(IntPtr c, String methodName, Object[] parameters, Type[] types, Object[] varargs, Type[] varargsTypes)
  at Microsoft.Dynamics.Ax.Xpp.XppObjectBase.Call(String methodName, Object[] parameters, Type[] types, Object[] varargs)
  at Microsoft.Dynamics.Ax.Xpp.DictTable.SupportInheritance()
  at Dynamics.AX.Application.SysDictTable.'getRootTable(Int32 _tabid) in xppSource://Source/ApplicationPlatform\AxClass_SYSDictTable.xpp:line 1498
  at Dynamics.AX.Application.SysDictTable.getTables(Int32 _tabid)
  at Dynamics.AX.Application.SysDataBaseLog.ConfigureSqlLogging() in xppSource://Source/ApplicationPlatform\AxTable_SYSDataBaseLog.xpp:line 60
  at Dynamics.AX.Application.SysDataBaseLog.ConfigureSqlLogging()
  at SysDataBaseLog::ConfigureSqlLogging(Object[], Boolean& )
  at Microsoft.Dynamics.Ax.Xpp.ReflectionCallHelper.MakeStaticCall(Type type, String MethodName, Object[] parameters)

DB sync failed.
```

**Reason:** This issue occurs because there is data in the SQL DatabaseLog table that conflicts with the metadata in the package.

**Resolution:** Run the following query on AXDB to clean the DatabaseLog table and retry the deployment.

```
select * into databaselog_bak from databaselog
truncate table databaselog
```

DBSync fails to start

**Issue:** During deployment, the deployment fails with the AXSF applications staying in "InBuild" status in Service Fabric explorer. When reviewing the logs on the AXSF nodes's work directories, the following DBSync error can be found.
Microsoft.Dynamics.AX.InitializationException: Database login failed. Please check SQL credentials and try again.
at Microsoft.Dynamics.AX.AOS.StartupInternal(String[] Arguments)
at Microsoft.Dynamics.AX.AOS.Startup()
at Microsoft.Dynamics.AX.AosConfig.?A0xb5100bbf.GetAosConfig()
at Microsoft.Dynamics.AX.AosConfig.Config.InitInternal()
at Microsoft.Dynamics.AX.AosConfig.Config.InitOnce(Boolean isOfflineMode)
at Microsoft.Dynamics.AX.Framework.Database.Tools.LegacyCodepath.StartAosCode(SyncOptions syncOptions, String sqlConnectionString)
at Microsoft.Dynamics.AX.Framework.Database.Tools.LegacyCodepath.ExecuteWithinAOS(SyncOptions syncOptions, String sqlConnectionString, IMetadataProvider metadataProvider, Func`1 func, Action`1 errorHandler)
at Microsoft.Dynamics.AX.Framework.Database.Tools.LegacyCodepath.NOTE_LeavingSynchronizer_CallStackAboveThisLineIsCustomCode(SyncOptions syncOptions, String sqlConnectionString, IMetadataProvider metadataProvider, Action`1 a)
at Microsoft.Dynamics.AX.Framework.Database.Tools.LegacyCodepath.RunCustomAction(SyncOptions syncOptions, String sqlConnectionString, IMetadataProvider metadataProvider, Action`1 a)
at Microsoft.Dynamics.AX.Framework.Database.Tools.SyncEngine.PreTableSync()
at Microsoft.Dynamics.AX.Framework.Database.Tools.SyncEngine.Run(String metadataDirectory, String sqlConnectionString, SyncOptions options)

Reason: This issue may occur because the SQL password contains special characters.

Resolution: Update the password of the SQL user and remove the special characters. Then, update the Credentials.json file with the new password and retry the deployment from LCS.

DBSync fails with PEAP and first release APP version 10.0.14 Platform update 38

Issue: During deployment, the deployment fails with the AXSF applications staying in "InBuild" status in Service Fabric explorer. When reviewing the logs on the AXSF nodes's work directories, the following DBSync error is present multiple times.

 Reason: This issue occurs because the SQL Server account used by Finance + Operations does not have sufficient permissions to execute the operation.

Resolution: Execute the following command in your SQL Server:

```
use master
GRANT ALTER ANY EVENT SESSION to axdbadmin;
```
ReportingService fails to start

**Issue:** During deployment, the ReportingService fails to start. In the event logs you will see the following error:

```csharp
Could not load file or assembly 'Microsoft.SqlServer.BatchParser, Version=12.0.0.0, Culture=neutral, PublicKeyToken=89845dcd8080cc91' or one of its dependencies. An attempt was made to load a program with an incorrect format.
System.Reflection.RuntimeAssembly.GetType(RuntimeAssembly assembly, String name, Boolean throwOnError, Boolean ignoreCase, ObjectHandleOnStack type)
at System.Reflection.RuntimeAssembly.GetType(String name, Boolean throwOnError, Boolean ignoreCase) at System.Reflection.Assembly.GetType(String name, Boolean throwOnError)
at Microsoft.SqlServer.Management.Common.ServerConnection.GetStatements(String query, ExecutionTypes executionType, Int32& statementsToReverse)
at Microsoft.SqlServer.Management.Common.ServerConnection.ExecuteNonQuery(String sqlCommand, ExecutionTypes executionType)
at Microsoft.Dynamics.AX.Framework.Reports.Setup.ReportsIdentityUpdater.ExecuteSQlScript(String script)
at Microsoft.Dynamics.AX.Framework.Reports.Setup.ReportsIdentityUpdater.CreateReportServerDatabase(String userName, SrsWmi rsconfigSetting)
at Microsoft.Dynamics.AX.Framework.Reports.Setup.ReportsServerInstaller.SetInstanceIdentity(String instanceName, String username)
at Microsoft.Dynamics.AX.Framework.Reports.Setup.RunReportsSetup.Execute(String path, String encodedConfigurationValues)
```

**Reason:** This issue occurs because the correct version of SSMS is not installed. The version of SSMS that should be installed is 17.9.1.

**Resolution:** Install SSMS version 17.9.1.

Report deployment fails on version 10.0.19 and later

**Issue:** During deployment, the report deployment operation fails. In the report deployment log, you will see the following error.

```csharp
Publish-AXReport : Value cannot be null.
Parameter name: The value supplied for parameter 'serviceName' cannot be null or empty.
At C:\ProgramData\SF\AOS12\Fabric\work\Applications\AXSFType_App110\AXSF.Scripts.1.0.20210617153432\Reporting.psm1:492 char:9
+    Publish-AXReport -MaxDegreeOfParallelism 1 -ErrorAction Conti ...
+    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
+ CategoryInfo          : OpenError: (Microsoft.Dynam...hReportCommand:PublishReportCommand) [Publish-
AXReport], ArgumentNullException
+ FullyQualifiedErrorId : Value cannot be null.
Parameter name: The value supplied for parameter 'serviceName' cannot be null or empty.
```

**Reason:** The AOS needs to retrieve the list of services running on the BI node to find the version of SSRS that is currently installed. The account that the AOS runs under does not have the appropriate permissions to get the list of services, so it fails and is unable to retrieve the serviceName.

**Resolution:** Version 2.11.1 of the infrastructure scripts, available from the Shared asset library in LCS, has been released to propagate these permissions so the serviceName can be retrieved.

**NOTE**

If you used 2.11.0 of the infrastructure scripts, download the newest version and go through the following steps again.

**Automatically add these permissions:**

- `Add-Member -TargetType `Microsoft.Dynamics.AX.Framework.Management.Reports.PublishReportCommand` -MemberType Property -Name `filename` -Value `file.name`
- `Add-Member -TargetType `Microsoft.Dynamics.AX.Framework.Management.Reports.PublishReportCommand` -MemberType Property -Name `folder` -Value `file.baseDirectory`
- `Add-Member -TargetType `Microsoft.Dynamics.AX.Framework.Management.Reports.PublishReportCommand` -MemberType Property -Name `environment` -Value `environment.name`
- `Add-Member -TargetType `Microsoft.Dynamics.AX.Framework.Management.Reports.PublishReportCommand` -MemberType Property -Name `password` -Value `file.password`
- `Add-Member -TargetType `Microsoft.Dynamics.AX.Framework.Management.Reports.PublishReportCommand` -MemberType Property -Name `script` -Value `file.text`
- `Add-Member -TargetType `Microsoft.Dynamics.AX.Framework.Management.Reports.PublishReportCommand` -MemberType Property -Name `serviceName` -Value `file.name`
1. Download the latest infrastructure scripts from the Shared asset library in LCS.

2. Migrate your ConfigTemplate.xml if needed.

3. Run the following commands in PowerShell with Administrator privileges:

   ```powershell
   .\Export-Scripts.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
   .\Export-PfxFiles.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
   ```

4. Copy the generated VM folder to the BI node if not using the remoting scripts.

5. Run the following command in PowerShell with Administrator privileges.

   ```powershell
   # If remoting, execute
   # .\Complete-PreReqs-AllVMs.ps1 -ConfigurationFilePath .\ConfigTemplate.xml -ForcePushLBDScripts
   .\Complete-PreReqs.ps1
   ```

6. Run the following command in PowerShell with Administrator privileges to verify the setup.

   ```powershell
   # If Remoting, execute
   # .\Test-D365FOConfiguration-AllVMs.ps1 -ConfigurationFilePath .\ConfigTemplate.xml
   .\Test-D365FOConfiguration.ps1
   ```

   **IMPORTANT**
   If you used remoting, be sure to run the cleanup steps after the setup is completed. For instructions, see Step 20. Tear down CredSSP, if remoting was used.

   Manually add these permissions:

   1. Go to your BI node.

   2. Open lusrmgr.msc (Local Users and Groups).

   3. Create a new group called **Dynamics365ReadServices**.

   4. Add the account that your AOS runs under (such as axserviceuser, svc-AXSF$) to the group you created above.

   5. Download the latest infrastructure scripts from the Shared asset library in LCS.

   6. Copy the infrastructure scripts to your BI (SSRS) node.

   7. Create a file scmgroups.csv with the following content.

      ```csv
      "Name"
      "Dynamics365ReadServices"
      ```

   8. Run the following command in PowerShell with Administrator privileges.

      ```powershell
      .\Set-ServiceControlManagerPermissions.ps1
      ```

   9. Run the following command in Powershell with Administrator privileges to verify the setup.

      ```powershell
      .\Set-ServiceControlManagerPermissions.ps1 -Test
      ```
This topic will serve as a central repository for scripts that you can use to fix issues in on-premises environments. These scripts must usually be run as pre-deployment or post-deployment scripts.

For more information about how to resolve issues in on-premises environments, see Troubleshoot on-premises deployments.

Prepare your environment for script execution

1. Configure the execution of pre-deployment and post-deployment scripts. For more information, see Local agent pre-deployment and post-deployment scripts.

2. Add the following code to your Predeployment.ps1 script.

```powershell
# This has to be filled out
# $agentShare = '<Agent-share path>' # E.g '\\LBDContosoShare\agent'
$agentShare = '\servername\D365FFOAgent'
Write-Output "AgentShare is set to $agentShare"

# The scripts make the assumption that the wp folder only contains one folder for the environment name.
# If you have multiple folders in there from older deployments, then please remove those.
# It is not recommended to use the same agent share for multiple environments.

#& $agentShare\scripts\TSG_UpdateFRDeployerConfig.ps1 -agentShare $agentShare
#& $agentShare\scripts\TSG_WindowsAzureStorage.ps1 -agentShare $agentShare
#& $agentShare\scripts\TSG_SysClassRunner.ps1 -agentShare $agentShare
#& $agentShare\scripts\TSG_RemoveFilesFromZip.ps1 -agentShare $agentShare -filesToRemove

#& $agentShare\scripts\TSG_EnableGMSAForAOS.ps1 -agentShare $agentShare -gmsaAccount contoso\svc-AXSF$
```

3. From the relevant section of this topic, copy the code that you require to fix your issue, and paste it into a new file. Save this file in the same folder where your Predeployment.ps1 script is stored. The file name must match the title of the section that you copied the code from. Repeat this step for other issues that you must fix.

4. In the Predeployment.ps1 script, in the code that you added earlier, uncomment the lines that invoke the scripts that you want to use.
The following script is used to fix an issue that occurs when SysClassRunner is run in some versions of the platform. For more information about this issue, see [SysClassRunner doesn't run successfully](#).
TSG_UpdateFRDeployerConfig.ps1

The following script is used to fix an issue that occurs when Financial Reporting is deployed in some versions of the platform. For more information about this issue, see Could not load file or assembly EntityFramework.
param (  
    [Parameter(Mandatory=$true)]  
    [string]  
    $agentShare = ''  
)  

$frConfig = Get-ChildItem $agentShare\wp\*\StandaloneSetup-  
  *\Apps\FR\Deployment\FinancialReportingDeployer.exe.config |  
  Select-Object -First 1 -Expand FullName  

if(-not $frConfig)  
{  
    Write-Output "Unable to find FinancialReportingDeployer.exe.Config"  
    return  
}  

Write-Output "Found config: $frConfig"  

$xml = get-content $frConfig  

$nodeList = $xml.GetElementsByTagName("loadFromRemoteSources")  

if($nodeList.Count -eq 0)  
{  
    # Create the node  
    $newNode = $xml.CreateNode("element","loadFromRemoteSources","")  
    $newNode.SetAttribute("enabled","true")  
    # Find the parent  
    $nodeList = $xml.GetElementsByTagName("runtime")  
    $runtimeNode = $nodeList[0]  
    $runtimeNode.AppendChild($newNode)  
    # Save doc  
    $xml.save($frConfig)  
    Write-Output "Inserted new node: "$newNode.Name  
}  
else  
{  
    $node = $nodeList[0]  
    $attribute = $node.Attributes.GetNamedItem("enabled")  
    if($attribute.Value -eq "true")  
    {  
        Write-Output "Node already exists: "$node.Name  
    }  
    else  
    {  
        Write-Output "Node already exists but attribute is incorrect: "$attribute.Name "is"  
        $attribute.Value  
    }  
}  

TSG_WindowsAzureStorage.ps1  

The following script is used to fix an issue where files can't be downloaded or exported in some versions of the platform.
TSG_RemoveFilesFromZip.ps1

The following script is used to fix an issue that occurs for some customers who were previously on versions 10.0.5 through 10.0.9. Due to how the prepare process works, there are some older versions of DLLs that remain in the TaxEngine folder, which in newer releases have been moved to different module folders. This script ensures that these DLLs are removed from the downloaded asset, before being deployed to the AOS nodes.
[CmdletBinding()]
param
  (  
    [Parameter(Mandatory)]
    [ValidateNotNullOrEmpty()]
    [ValidateScript({ Test-Path -Path $_ })]
    [string] $agentShare,
    
    [ValidateNotNullOrEmpty()]
    [string[]]$filesToRemove
  )

#requires -Version 5
$errorActionPreference = 'Stop'
$infoPreference = 'Continue'
$files = @() [Reflection.Assembly]::LoadWithPartialName('System.IO.Compression')

if(! (Test-Path $AgentShare))
{
  throw "The path provided for the agentshare is not valid/accessible."
}

ForEach($file in $FilesToRemove)
{
  if(!$file.StartsWith('Packages'))
  {
    throw "The path of $file does not start with Packages."
  }
  $files += $file.Replace('\\', '/')
}

$basePath = Get-ChildItem $AgentShare\wp\*\StandaloneSetup-*\Apps | Select-Object -First 1 -Expand Fullname

#Some customers experience an unexpected behavior with the previous command.
if($basePath -notmatch "\AOS")
{
  $basePath = Join-Path $basePath -ChildPath "\AOS"
}

$basePath = Join-Path $basePath -ChildPath "AXServiceApp\AXSF\Code"
$zipfile = Join-Path $basePath -ChildPath "Packages.zip"

try
{
  $stream = New-Object IO.FileStream($zipfile, [IO.FileMode]::Open)
  $mode   = [IO.Compression.ZipArchiveMode]::Update
  $zip    = New-Object IO.Compression.ZipArchive($stream, $mode)

  ($zip.Entries | ? { $files -contains $_.FullName }) | % { $_.Delete() } | Write-Host
}
catch
{
  throw 'An Error Occurred'
}
finally
{
  $zip.Dispose()
  $stream.Close()
  $stream.Dispose()
}
The following script is used to change the account the AOS runs under from an Active Directory (AD) user to a group Managed Service Account (gMSA).

```
param (        
    [Parameter(Mandatory)]        
    [ValidateNotNullOrEmpty()]    
    [ValidateScript({ Test-Path -Path $_ })]    
    [string] $agentShare,        

    [Parameter(Mandatory=$true)]    
    [string] $gmsaAccount        
)

$basePath = Get-ChildItem $agentShare\wp\*\StandaloneSetup-*\ | Select-Object -First 1 -Expand FullName
if(!(Test-Path $basePath)) {
    Write-Error "Basepath: $basePath , not found" -Exception InvalidOperation
}

$configJsonPath = "$basePath\config.json"
$configJson = Get-Content $configJsonPath | ConvertFrom-Json
$updatedComponents = @()
foreach ($component in $configJson.components) {
    if($component.name -eq "AOS") {
        $component.parameters.infrastructure.principalUserAccountType.value = "ManagedServiceAccount"
        $component.parameters.infrastructure.principalUserAccountName.value = $gmsaAccount
    }
    if($component.name -eq "ReportingServices") {
        $component.parameters.accountListToAccessReports.value = $gmsaAccount
    }
    $updatedComponents += $component
}
$configJson.components = $updatedComponents
$configJson | ConvertTo-Json -Depth 100 | Out-File $configJsonPath
Write-Host "Successfully updated the configuration for AOS gMSA execution."
```

NOTE
This script can only be used starting with version 10.0.17. You will need to reinstall the printers on each AOS node as they are not available to the gMSA account. For more information, see Install network printer devices in on-premises environments.
An AOT package is a deployment and compilation unit of one or more models that can be applied to an environment. It includes model metadata, binaries, reports and other associated resources. One or more AOT packages can be packaged into a deployable package, which is the vehicle used for deployment of code (and customizations) on demo, sandbox, and production environments. This topic guides you through the process of creating and applying a deployable package.

Overview of the process

In order to deploy your code and customizations to a runtime environment (demo, sandbox, or production), you must create deployable packages of your solution or implementation. Deployable packages can be created by using Visual Studio dev tools or by using the build automation process that is available on build environments. These deployable packages are referred to as Application Deployable Packages or AOT Deployable Packages. The following image shows an overview of the process. After a deployable package is created, it must be uploaded to the Lifecycle Services (LCS) project's asset library. An administrator can then go to the LCS environment page and apply the package to a runtime environment using the Maintain > Apply updates tool.

NOTE
Custom payment connector for Commerce needs to be packaged using a combined AOT deployable package. For more information, see Create payment packaging for Application Explorer in Service Fabric deployments.

NOTE
Application Deployable Packages do not contain source code.

It is always recommended to use a build environment to create deployable packages that are intended to go to production.

Create a deployable package

We recommend using a build environment to create deployable packages. You can also create a deployable package on a development environment.
On a development environment, after you have completed development and testing, follow these steps to create a deployable package in Visual Studio.

1. In Microsoft Visual Studio, select Dynamics 365 > Deploy > Create Deployment Package.

![Create Deployment Package](image)

2. Select the packages that contain your models, and then select a location in which to create the deployable package.

![Select Packages](image)

3. After a deployable package is created, sign in to Lifecycle Services, and then, in your LCS project, click the Asset Library tile.

4. Upload the deployable package that you created earlier.
Apply a deployable package
To apply a deployable package to an environment, see Apply updates to cloud environments.

Remove a deployable package
To uninstall or remove a deployable package from an environment, see Uninstall a package.
This topic walks you through the steps for using the command line to apply either a binary update or an application (AOT) deployable package that was created in your development or build environment.

**IMPORTANT**
For most types of environments, you can apply a deployable package to an environment directly from Microsoft Dynamics Lifecycle Services (LCS). For more information, see [Apply updates to cloud environments](#). Therefore, this topic applies primarily to environment types that don't support the application of updates via LCS. Examples include local development environments (downloadable virtual hard disks [VHDs]), multi-box development/test environments in Microsoft Azure (LCS Partner and trial projects), and build environments. However, you can also use this topic any time that you want to install deployable packages by using the command line instead of LCS.

**Key concepts**
- **Deployable package** – A deployable package is a unit of deployment that can be applied to any environment. It can consist of a binary hotfix to the runtime components of Application Object Server (AOS), an updated application package, or a new application package.
- **AXUpdateInstaller** – AXUpdateInstaller is an executable program that is bundled in the deployable package. When the package is downloaded to a computer, the installer is available.
- **Runbook** – The deployment runbook is a series of steps that is generated and used to apply the deployable package to the target environment. Some of the steps are automated, and some are manual. AXUpdateInstaller enables these steps to be run one at a time and in the correct order.

**Install an application (AOT) deployable package on a development environment**

**NOTE**
The steps listed below are for customization packages only. Do not use the `devinstall` parameter when running the Data Upgrade deployable package as part of an upgrade from Microsoft Dynamics AX 2012 to a Finance and Operations app.

An AOT deployable package is a package that contains customizations and extensions to your application. If you want to use the command line just to install an AOT deployable package on a development or demo environment, follow the instructions in this section. You can then skip the rest of this topic.

1. On the virtual machine (VM), download the zip file for the deployable package. Make sure that the zip file is stored in a non-user folder.

   **NOTE**
   After you download the zip file, right-click it, and then select Properties. Then, in the Properties dialog box, on the General tab, select Unblock to unlock the files.

2. Extract the files.
3. Open a Command Prompt window, and go to the folder where you extracted the deployable package.

4. Run the following command.

```
AXUpdateInstaller.exe devinstall
```

The **devinstall** option installs the AOT deployable package on the VM.

**NOTE**

This command doesn't run database synchronization. You must run database synchronization from Microsoft Visual Studio after you install the deployable package.

---

### Collect topology configuration data

1. In LCS, on the **Environment** page, select the name of a VM. Establish a Remote Desktop connection to the VM by using the user name and password that are provided on the **Environment** page.

2. On the VM, download the zip file for the deployable package from LCS. Make sure that the zip file is stored in a non-user folder.

**NOTE**

After you download the zip file, right-click it, and then select **Properties**. Then, in the **Properties** dialog box, on the **General** tab, select **Unblock** to unlock the files.

3. Extract the files.

4. In the folder where you extracted the deployable package, find and open the file that is named **DefaultTopologyData.xml**. You must specify the VM name and the installed components in this file.

- To specify the VM name, follow these steps:
  
  a. In File Explorer, right-click **This PC**, and then select **Properties**.
  
  b. In the system properties, find and make a note of the computer name (for example, **AOS-950ed2c3e7b**).
  
  c. In the **DefaultTopologyData.xml** file, replace the machine name with the computer name that you found in the previous step.

- To specify the installed components, follow these steps:

  a. Open a Command Prompt window as an administrator.

  b. Go to the extracted folder, and run the following command to see a list of all the components that are installed on the computer.

```
AXUpdateInstaller.exe list
```

  c. Update the **DefaultTopologyData.xml** file with the list of components.

When you've finished specifying the VM name and the installed components, the **DefaultTopologyData.xml** file should resemble the following illustration.
Generate a runbook from the topology

Based on the topology information in the DefaultTopologyData.xml file, you must generate the runbook file that will provide step-by-step instructions for updating each VM.

- On any VM, run the following command to generate the runbook.

```
```

Here is an explanation of the parameters that are used in this command:

- **[runbookID]** – A parameter that is specified by the developer who applies the deployable package.
- **[runbookFile]** – The name of the runbook file to generate (for example, AOSRunbook.xml).

**Example**

```
AXUpdateInstaller.exe generate -runbookid="VAL200AA2BMEDIU-runbook" -
topologyfile="DefaultTopologyData.xml" -servicemodelfile="DefaultServiceModelData.xml" -runbookfile="VAL200AA2BMEDIU-runbook.xml"
```

The runbook provides the sequence of steps that must be run to update the environment. The following illustration shows an example of a runbook file. Each step in a runbook is associated with an ID, a machine name, and step execution details.
Install a deployable package

1. On the first machine (VM) that is listed in the runbook file, follow these steps:
   a. Import the runbook by running the following command.

      AXUpdateInstaller.exe import -runbookfile=[runbookFile]

      **Example**

      AXUpdateInstaller.exe import -runbookfile="VAL200AA2BMEDIU-runbook.xml"

   b. Verify the runbook.

      AXUpdateInstaller.exe list

   c. Run the runbook.

      AXUpdateInstaller.exe execute -runbookid=[runbookID]

      **Example**

      AXUpdateInstaller.exe execute -runbookid="VAL200AA2BMEDIU-runbook"

AXUpdateInstaller updates the runbook file after each step is run on a VM. The runbook also logs information about each step.

For manual steps, follow the instructions, and then run the following command to mark the step as completed in the runbook.
Verify installation

1. Run the following command to verify that the new updates are installed.

   AXUpdateInstaller.exe list

2. View the runbook to see the completed steps. Here is an example of a runbook file where the steps have been completed.

Example

AXUpdateInstaller.exe execute -runbookID=[runbookID] -setstepcomplete=[stepID]

AXUpdateInstaller.exe execute -runbookid="VAL200AA2BMEDIU-runbook" -setstepcomplete=2

If errors occur during any step, debug the script or the instructions in the step, and update accordingly.

d. Export the runbook.

   AXUpdateInstaller.exe export -runbookid=[runbookID] -runbookfile=[runbookFile]

Example

AXUpdateInstaller.exe export -runbookid="VAL200AA2BMEDIU-runbook" -runbookfile="VAL200AA2BMEDIU-runbook.xml"

2. Repeat step 1 on every other VM that is listed in the runbook file. For one-box environments, such as development, build, and demo environments, there is only one VM.
Backup the runbook file

- After all the steps in the runbook are completed and you’ve exported the runbook, save the file outside the computer for future reference. For example, you might have to use the runbook file in these situations:
  - You must analyze the downtime requirements for production, and so on.
  - You must send the file to Microsoft because a deployable package can’t be installed.

Troubleshooting

- If any step in the runbook fails, you can rerun it by running the following command.

  `AXUpdateInstaller.exe execute -runbookid=[runbookID] -rerunstep=[stepID]`

- To prevent version mismatch or downgrade, or installation of the same deployable package, run the following command.

  `AXUpdateInstaller.exe execute -runbookid=[runbook ID] -versioncheck=true`

- To verify database synchronization, in the `aosservice\scripts\` folder, find and open the `dbsync.error.txt` file, and look for any errors.
Occasionally, you might have to uninstall a deployable package. For example, you might be reorganizing your source code. Alternatively, you no longer require an independent software vendor (ISV) product and haven't renewed the license. Therefore, you must remove the package.

**Remove a model**

A model is a design-time concept that is part of a package. When a model isn't the only model in a module, you can just remove it from the source code. No other steps are required, because when you deploy the updated module, the old module is overwritten. All overlayer models fall into this category. For more information, see Deleting a model.

**Prerequisites**

- If any models reference the module that will be removed, the references must be removed from them. For information about how to find the references that must be removed, see Viewing model dependencies.
- Build and deploy any modules that references were removed from.
- All references to and from the modules must be removed before you begin to uninstall the module. To remove all a module's references, add a single class to the model. This class should contain no code. It should contain only a reference to the application platform.
- A Microsoft module cannot be removed. If this is attempted, a validation error will be shown on the package in Lifecycle Services.
- A module cannot be removed if it is part of the AOT deployable package being installed. If you want to remove a module, be sure that it is not part of the package before adding the name to the ModuleToRemove.txt file.

**Uninstall a package**

1. Create a file that is named ModuleToRemove.txt.
2. In the file, put the name of each module that you want to remove on a separate line. Make sure that you've completed the prerequisites for each module that you're removing.
3. Create a valid deployable package, and put the ModuleToRemove.txt file in the package\AOSService\Scripts folder.
4. Upload the package to the Lifecycle Services asset library. Wait for the asset to finish validation, and review any warnings that are shown on the Asset Details panel on the right side of the page.
5. Install the deployable package. For more information about how to install deployable packages, see Apply updates to cloud environments.
6. Verify that the package was uninstalled before you complete this procedure in a production environment.
This topic provides detailed information that will help you troubleshoot issues that might occur when you apply packages on your Tier 1 or Tier 2 through Tier 5 environments. For information about how to apply a package, see Apply updates to cloud environments.

General troubleshooting and diagnostics

If a package isn't successfully applied, you have two options:

- Retry the operation that failed.
- Use the logs.

**Retry the failed operation**

If package application fails, and you want to retry the operation, select Resume.

**Use the logs**

If package application fails, and you want to use the logs, follow these steps.

1. Download and then unzip the log files.
2. Select the role that a step failed for, such as AOS or BI.
3. Select the virtual machine (VM) where the step failed. You can find this information in the **Machine name** column in the Environment updates section.
4. In the VM logs, select the folder that corresponds to the step where the issue occurred. The folder name identifies the step that each folder corresponds to.

   For example, if the issue occurred during the execution of a step, select the **ExecuteRunbook** folder. The step number is highlighted and is the number after the globally unique identifier (GUID).

Package application issues

**Issue: The package that was applied isn't valid**

**Description**

Because the package that was applied wasn’t valid, the servicing status is Failed, and no updates are listed in the Environment updates section. To verify whether the package is valid, follow these steps.

1. Download and unzip the logs.
2. Navigate to the logs for the Application Object Server (AOS) machine.
3. Verify that the **DownloadFilesAndSlipstreamTools-xxx** and **GenerateRunbook-xxx** folders exist.
4. Open the **GenerateRunbook-xxx folder**, and then open the file for the output type.

   If you find an error message or an exception that states that a file is missing or failed to generate any steps for the runbook, there was an attempt to upload a package that wasn’t valid.

**Action**

Select Abort to abort the current package, upload a new package, and then restart the servicing flow.
**Issue: Package deployment fails even though no steps failed**

**Description**

A time-out occurs when you download the package to the machine. During pre-servicing, a few steps must be completed before steps are completed in the runbook. As part of the pre-servicing, the package must be downloaded to all the machines. The time to download might vary slightly, depending on the datacenter where the environment resides. Any download doesn't occur within 30 minutes is considered a failure in the servicing status and will be stopped.

**Action**

1. Select **Resume** to see whether you can resolve the issue. If this step doesn't work, move on to step 2.
2. Download the logs from the **Environment** page.
3. Verify that the package download on all the machines has logs that include the `DownloadFilesAndSlipstreamTools` folder.
4. Review the log files. If you don't see the following text at the end of the log file, the issue occurred because the package download wasn't completed: “Completed file download and slipstream successfully”

**Issue: Package deployment fails even though no steps failed**

**Description**

There isn't enough disk space to download the package. Inspect all the machines. This issue occurs if the servicing drive of any machines is full.

Note that if you select **Resume** in this situation, you won't resolve the issue.

To verify that the deployment failed because space issues, follow these steps.

1. Navigate to the `DownloadFilesAndSlipstreamTools-xxx` folder.
2. Open the output file, and view the error message to see whether the step failed because space issues.

**Action**

As part of the automated cleanup on topologies, any deployable packages in `%ServiceVolume%\DeployablePackages` that are more than 30 days old are deleted. The same timeline is also used to delete servicing-related logs. These logs are usually located in `C:\Dynamics`.

However, on dev/one-box machines, there is more flexibility. The number of retention days and the minimum disk space of the ServiceVolume and Logs drives can be customized.

- Under the `HKLM:\SOFTWARE\Microsoft\Dynamics\Deployment` registry key, you can create the following keys to customize when cleanup should occur. The automated cleanup task will consider these values.
  - **CutoffDaysForCleanup** – The number of days that old packages and logs should be retained. The default value is 30.
  - **CutoffDiskSpaceLimitForPackages** – The minimum free disk space (in gigabytes [GB]) on the service volume drive where the package folder is located. For example, if the disk space is 200 GB, the cleanup task will remove the packages, based on the number of days.
  - **CutoffDiskSpaceLimitForLogs** – The minimum free disk space (in GB) of the system drive where the log folder is located. For example, if the disk space is 100 GB, the cleanup task will remove the servicing-related logs, based on the number of days.

**Issue: A step failed with errors**

**Description**

A step might fail with errors for one of the following reasons:
- The package has customization issues or is missing dependencies.
- There is an issue with the servicing scripts.
- A random failure occurred when the step was executed.

To verify what the issue is, follow these steps.

1. Download and navigate to the step logs.
2. Open the output file for the step, and see whether there are any errors.
3. If any additional log files are available, inspect them for errors.

**Action**

1. Download the logs.
2. Select **Rerun step** to retry the failed step.

If the step fails again, and the same error occurs, go back to the logs to look for more information.

- If you notice that there is an issue with the customizations, abort this package, and retry by using the new package.
- See whether a fix for the issue is available in Issue search.
- If you see the following step failure, either database synchronization or report deployment might have failed: "GlobalUpdate script for service model: AOSService"
- Look for the DBSync.err file, and see what the errors are. Inspect the DBSync.log file. For specific failures during the DB Sync step, look in the **Common DB Sync Failures** section.

**Issue: The deployment status is Servicing but the servicing status is Failed**

**Description**

A built-in mechanism enables the system to retry multiple times before it gives up. The following preparation steps can also be retried several times:

- Download the package to the machines.
- Slipstream the servicing package.
- Generate the runbook.

**Action**

You can download the logs to view the error and take action before all retries are exhausted. If the retry mechanism fails to go beyond the preparation stage, and you see a status of **Failed**, open a ticket so that the Microsoft team can investigate the issue further.

**Issue: The dashboard doesn't open after package deployment is completed**

**Description**

If the dashboard doesn't open after package deployment is completed, a run-time error might be occurring when the AOS machine is started.

**Action**

1. Start Event Viewer.
2. Navigate to **Applications and Services Logs > Microsoft > Dynamics > Ax-SystemRuntime > Operational Filter by errors**, see whether there are any errors, and investigate the errors as required.

**Issue: A package application failed with error code : DSU####**

**Description**

You may receive an error stating **A critical component has encountered an error processing your**
**Action**

Microsoft is proactively monitoring the service status and this type of outage is expected to be mitigated shortly.

There’s no impact on the status and the health of your environment.

If you experience this error when scheduling a service request or performing a database movement task, please try again at a later time.

**Typical database synchronization issues**

If you see the following step failure, a database synchronization issue might be occurring: "GlobalUpdate script for service model: A OSService"

Follow these steps to look for the DBSync.err file, find the errors, and inspect the DBSync.log file.

1. Download and navigate to the step logs.
2. Open the output file for the step, and see whether there are any errors.
3. If additional log files are available, inspect the logs for any errors.
4. The following table shows the failures that are seen most often and the action that you should take. The **Issue** column shows the error message that you will see in the dbsync.err file. The percentage sign (%) can be replaced with the metadata name of the table, field, index, and so on.

<table>
<thead>
<tr>
<th>ISSUE</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>%Table Sync Failed for Table%Create Unique Index%</td>
<td>This issue typically occurs when a unique index is created, but the data isn't unique. Fix the data before you run the step again.</td>
</tr>
<tr>
<td>%Application configuration sync failed.%Custom action sync failed with error:%</td>
<td>View the information in the error message and the call stack to determine the application code that is causing the issue.</td>
</tr>
<tr>
<td>%cannot be found from underlying query’s table%</td>
<td>This issue was fixed. For more information, refer to KB 4018815.</td>
</tr>
<tr>
<td>%Table Sync Failed for Table%Converting Field%</td>
<td>Follow the error message, fix the issue, and run the step again.</td>
</tr>
<tr>
<td>%failed because one or more objects access this column%</td>
<td>See whether the index is in the metadata. If the index is in the metadata, this issue is a SyncEngine product issue. If the index isn’t in the metadata, remove the index from the SQL database before you run the step again.</td>
</tr>
<tr>
<td>%cannot be found from underlying data source’s table%</td>
<td>This issue was fixed. For more information, refer to KB 4018815.</td>
</tr>
<tr>
<td>‘%Table Sync Failed for Table%’ and errorMessage like ‘%There is already an object named%’</td>
<td>The Internal SqlDictionary table and SQL schema are out of sync. There isn’t enough information in the logs to understand how this state was reached.</td>
</tr>
<tr>
<td>ISSUE</td>
<td>ACTION</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>%Table Sync Failed for Table%Column names in each table must be unique%</td>
<td>SqlDictionary entries for the table are corrupted, and the field is missing. There isn't enough information in the logs to understand how this state was reached.</td>
</tr>
<tr>
<td>%Column name 'LOAD_' does not exist in the target table or view. CREATE INDEX%</td>
<td>This issue appears to be a SyncEngine issue. Create a ticket on Microsoft Support.</td>
</tr>
<tr>
<td>Cannot drop the index 'VENDREQUESTPROFILEQUESTIONNAIRE.I_1301PROFILEQUESTIONNAIRE', because it does not exist or you do not have permission</td>
<td>This issue appears to be a SyncEngine issue. Create a ticket on Microsoft Support.</td>
</tr>
<tr>
<td>%The index entry of length 2046 bytes for the index 'I_65750INDEX1' exceeds the maximum length of 1700 bytes for nonclustered indexes%</td>
<td>Modify the index before you run the step again.</td>
</tr>
<tr>
<td>%Incorrect syntax near%</td>
<td>This issue is a SyncEngine issue. Create a ticket on Microsoft Support.</td>
</tr>
<tr>
<td>Reference to database and/or server name in 'TEMPDB.DBO.T_TRVREQUISITIONLINE_C4C3569DD5A1A4CDABAE71A341743FB61' is not supported in this version of SQL Server</td>
<td>This issue is a SyncEngine issue. Create a ticket on Microsoft Support.</td>
</tr>
<tr>
<td>%Error: Timeout expired. The timeout period elapsed prior to obtaining a connection from the pool%</td>
<td>Retry the step.</td>
</tr>
<tr>
<td>Database execution failed: Invalid column name 'DEFAULTDIMENSION'. CREATE VIEW</td>
<td>This issue appears to be a SyncEngine issue. Create a ticket on Microsoft Support.</td>
</tr>
<tr>
<td>Database execution failed: Invalid object name 'PMBI_DEPROJECTTIMESHEET'. CREATE VIEW</td>
<td>This issue appears to be a SyncEngine issue. Create a ticket on Microsoft Support.</td>
</tr>
<tr>
<td>%provider: Named Pipes Provider, error: 40 - Could not open a connection to SQL Server%</td>
<td>This issue should have been fixed in Platform Update 3. Retry the step.</td>
</tr>
</tbody>
</table>
Finance and Operations apps are often customized and extended to fit an organization's needs. If your solution is based on Microsoft Dynamics 365 Finance, Dynamics 365 Supply Chain Management, or Dynamics 365 Commerce, you can connect solution-specific and customer-specific Help content to the Help pane in the Finance and Operations client. This topic describes the main steps and decision points.

NOTE

Users of Finance and Operations apps can create custom task guides to supplement conceptual content that describes the functionality of their solution. These conceptual descriptions are also referred to as Help and can be provided by Microsoft, partners, and an organization itself. For more information, see Help system.

The following illustration, and this topic in general, use the term Help for conceptual descriptions that either include or exclude how-to guides. The term task guides refers to in-product task guides.

Custom Help content

Custom Help content typically originates from one of three sources:

- Microsoft documentation repositories (repos)

You can use the HTMLFromRepoGenerator tool from the Custom Help Toolkit to clone content from any of the Finance and Operations repositories and generate corresponding HTML files. Those files can then be updated with content that is specific to your solution.
• Existing customized Dynamics AX content

You can convert Dynamics AX custom Help content so that it can be used in Dynamics 365.

• HTML files that are created specifically for your solution

Learn more about the metadata that must be added to your HTML files for context-sensitive Help and search to work correctly.

Process

The end-to-end process depends on the actual customer solution and the users’ expectations. A typical process involves the following steps:

1. Create the custom Help content.
2. Publish the content on a website.
3. Index the content by using a search service.
4. Connect the custom Help pane to the website and the search service.

Microsoft provides a toolkit that can help you generate HTML files from the Microsoft Help repositories, generate JavaScript Object Notation (JSON) files for search services, and change the locale of HTML files so that it matches the locale of your solution.

You're welcome to share your knowledge by contributing to this documentation through the link at the bottom of the page or by joining the Dynamics 365 community.

The following table outlines the main objectives that admins typically have for configuring the Help experience.

<table>
<thead>
<tr>
<th>OBJECTIVE</th>
<th>LEARN MORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>I want to give my users a customized in-product Help experience that reflects their actual solution.</td>
<td>See the Custom Help websites section of this topic and Create documentation or training with Task Recorder.</td>
</tr>
<tr>
<td>I want to use the Microsoft Help content as a baseline for Help content that is specific to my solution.</td>
<td>See Custom Help Toolkit: The HtmlFromRepoGenerator tool.</td>
</tr>
<tr>
<td>I want to contribute to the Microsoft Help content.</td>
<td>See Extend, customize, and collaborate on the Help.</td>
</tr>
<tr>
<td>I want to reuse my existing Dynamics AX content.</td>
<td>See Convert Dynamics AX custom Help for use in Dynamics 365.</td>
</tr>
<tr>
<td>I want to set up a website for my Help content.</td>
<td>See the Custom Help websites section of this topic.</td>
</tr>
<tr>
<td>I want to add my content to the Help pane.</td>
<td>See Connect a custom Help website to the Help pane.</td>
</tr>
<tr>
<td>Our technical writers want guidance that will help them convert our earlier content into Markdown so that it becomes easier for them to customize the Microsoft content.</td>
<td>See Moving to Markdown.</td>
</tr>
</tbody>
</table>

Custom Help websites

Before the product can connect to your Help content, you must customize the in-product Help pane so that it shows your content. The following conditions must be met:

• Your content must be available on a website.

You can deploy your content to an existing website, or you can set up a dedicated website to host your
content. The website can be private or public, but we recommend that users **not** be required to sign in to access your content.

- Your content must be indexed by a search service.

  If you use the AzureSearchCustomHelp solution that is part of the Custom Help Toolkit for context-sensitive Help, the Help pane will generate a query that must be run against the search service's index. The query depends on specific metadata in the Help topics. For more information, see Metadata requirements for custom Help topics.

The Deploy custom Help to Azure topic describes an approach for hosting content on Azure. It includes information about how to set up a search service that indexes your content so that it can be found by the in-product Help pane. If you don't have an Azure subscription, create an account before you begin. You can start with a free account for 12 months. For more information, see Create your Azure free account today.

**See also**

- Connect a custom Help website to the Help pane
- Deploy custom Help to Azure
- Custom Help Toolkit
- Language and locale descriptors in the product and in Help
- Configure the Help experience for Finance and Operations apps
- Help system
Content that can be used with the Help pane in Microsoft Dynamics 365 Finance, Dynamics 365 Supply Chain Management, and Dynamics 365 Commerce can be derived from existing Microsoft content, migrated from existing Dynamics AX 2012 Help content, or created as new files.

Microsoft creates Help in multiple languages for the locales that are supported by Finance, Supply Chain Management, and Commerce. However, locale support isn’t restricted to those locales.

Creating custom Help content that is derived from existing Microsoft content

You can use Microsoft Help content as a baseline for content that describes your solution. The HtmlFromRepoGenerator tool can retrieve the content from Markdown files in Microsoft repositories (repos) and convert it to HTML files.

For more information about how to use existing Microsoft content as a baseline for content that describes your solution, see Extend, customize, collaborate on the Help.

Migrating content from existing AX 2012 Help content

If you have existing content from AX 2012, you can reuse it for Finance, Supply Chain Management, and Commerce. However, you must transform the HTML files so that they can be used in the custom Help environment. The Custom Help Toolkit includes a Windows PowerShell script, run_ax2012.ps1, that transforms AX 2012 HTML files so that they can be used in the custom Help environment.

Creating new Help content

You use the AzureSearchCustomHelp solution that is provided as part of the Custom Help Toolkit to connect your content to the Help pane. The Help pane will generate a query that is run against the search service’s index. Context-sensitive Help and full-text search in the AzureSearchCustomHelp solution require that each topic contain specific metadata.

**Metadata requirements for custom Help topics**

The following metadata must be present in your topics for context-sensitive Help and full-text search to return results in the AzureSearchCustomHelp solution.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>The value is used for full-text search from the Help pane.</td>
</tr>
<tr>
<td>description</td>
<td>The value is used for full-text search from the Help pane.</td>
</tr>
<tr>
<td>ms.search.form</td>
<td>The value contains the Application Object Tree (AOT) name of a page and is used for context-sensitive search from the Help pane.</td>
</tr>
</tbody>
</table>
The value indicates the language of the topic. It's mapped against the current browser locale when the Help pane searches the content. Language fallback can be configured for the target custom Help website. For more information, see Language and locale descriptors in the product and in Help.

The value determines which client the Help topic is shown in. You can specify one or more values. Values include Core, Operations, Retail, and Human Resources.

The following table describes the values that can be specified for the `ms.search.scope` property. You can specify one or more values. The values determine which client a Help topic is shown in.

<table>
<thead>
<tr>
<th>VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>If this value is present, the topic appears in the Help pane. Otherwise, the topic doesn't appear in the Help pane. This value is set for the part of the Microsoft content that must always be available in the Help pane, for all users across all supported Dynamics 365 solutions.</td>
</tr>
<tr>
<td>Operations</td>
<td>This value applies to solutions that are based on Finance or Supply Chain Management.</td>
</tr>
<tr>
<td>Retail</td>
<td>This value applies to solutions that are based on Commerce.</td>
</tr>
<tr>
<td>Human Resources</td>
<td>This value applies to solutions that are based on Dynamics 365 Human Resources.</td>
</tr>
<tr>
<td>Talent</td>
<td>This value applies to solutions based on Dynamics 365 Talent. (Note that the Dynamics 365 Talent: Attract and Dynamics 365 Talent: Onboard apps are being retired. For more information, see Retiring Dynamics 365 Talent: Attract and Onboard apps.)</td>
</tr>
</tbody>
</table>

Non-required metadata

The following properties are reserved for future use:

- `ms.search.region` – Eventually, this property might be used to limit the content that is shown to content that is tagged either as global or for the region of the implementation.
- `ms.search.validFrom` – Eventually, this property might be used to limit the content that is shown to content for a product that was released on a given date or earlier.
- `ms dyn365.ops.version` – Eventually, this property might be used to limit the content that is shown to content for a specific version of a product or earlier.
- `ms.search.industry` – Eventually, this property might be used to limit the content that is shown to content for a specific industry.

TIP

Microsoft content in the public GitHub repos contains additional metadata that Microsoft uses in internal processes that aren't related to the mechanics of the Help system. You can ignore these metadata properties if you extend or customize the Microsoft content.
Changing the locale of topics to match the locale of solutions

If your solution is intended to support multiple markets, you will want to provide Help content for each market. For example, your solution might support German (Germany) and German (Austria), but you have HTML files only for German (Germany). To make the same content available in German (Austria), you can make a copy of the HTML files and then use the HtmlLocaleChanger tool to update the ms.locale metadata. You can also add content that is specific to Austria to these new HTML files, as required.

Converting HTML files to JSON files for use with an Azure search service

If you use the AzureSearchCustomHelp solution that is provided as part of the Custom Help Toolkit, the search service requires that all your content be in JavaScript Object Notation (JSON) format. The ConvertHtmlToJson tool transforms HTML files into JSON files.

See also

Custom Help overview
Custom Help Toolkit
Language and locale descriptors in the product and in Help
Configure the Help experience for Finance and Operations apps
Help system
This topic describes the steps for setting up a web app to host your content and for setting up a search service to make the content discoverable by the in-product Help pane. You will set up a Microsoft Azure web app and then use that web app to host content that is connected to the in-product Help pane.

If you don't have an Azure subscription, create an account before you follow the steps in this topic. You can start with a free account for 12 months. For more information, see Create your Azure free account today.

Get started

The Prepare content for use with the Help pane topic describes the steps for preparing Help content so that it can be used with the in-product Help pane. After you have a set of HTML files and their equivalent JavaScript Object Notation (JSON) files, you can set up the web app and the search service.

Process overview

The general process for creating your Azure resources consists of the following steps:

1. In the Azure portal, create a resource group.
2. In the Azure portal, create a web app, a storage account, and a search service.

   - The web app stores and serves HTML files. The HTML files contain your Help content.
   - The storage account uses a blob container to store JSON files. The JSON files are your Help files after they have been converted to JSON format so that they can be used to generate an index of your content for search purposes. For more information, see Custom Help Toolkit: The ConvertHtmlToJson tool.
   - The search service indexes the Help content. An index makes your content discoverable by the in-product Help pane. For more information, see Create a basic index in Azure Cognitive Search.

   When you complete this step, you put the HTML files in the relevant language subfolders, based on the locales that the content was written for. For information about the names to use for these subfolders, see Language and locale descriptors in the product and in Help.
4. Upload JSON files into Azure Blob storage in the storage container, in subfolders that correspond to the language subfolders for the HTML files.
5. Configure the search service so that it has a data source, index, and indexer on the search service, by using the REST application programming interface (API).

   In this example, an API tool that is named Postman is used to make the REST API calls. To use a language-specific index analyzer, you must create language-specific indexes.

In the remaining sections of this topic, the assumption is that you have an Azure account and a valid subscription. If you don’t have an Azure subscription, create an account before you begin. You can start with a free account for 12 months. For more information, see Create your Azure free account today.

Create a resource group

To host your web app, search service, and storage account, you must first create one or more resource groups. We recommend that you create all resources in a single resource group to make management easier. For more
Create a web app

To host your content, you must create a web app in Azure. For more information, see Create a static HTML web app in Azure.

Create the web app

1. In the Azure portal, select Resource groups, select Add, and then specify a name for the resource group, such as MyCustomHelp.

2. Select Review + Create to finish creating the resource group.

Upload HTML files

1. In the Azure portal, go to your resource group, select Add, select Web App, and then specify the runtime stack and a name for the web app, such as MyCustomHelpWebApp.

   You can use any .NET Core stack as the runtime stack. For more information, see Create an ASP.NET Core web app in Azure.

2. After the deployment is completed, select Go to resource.

3. On the left side of the page, select Deployment Center. Under Manual Deployment (push/sync), select FTP, and then select Dashboard.

   You can use either app credentials or user credentials to upload your content to the web app. We recommend that you use app credentials. To upload your content, you must have the FTP/FTPS endpoint, the user name, and the password.

   You might want to copy the user name and password to a temporary location before you continue. Additionally, we recommend that you reset the credentials after you've completed the deployment. For more information, see Configure deployment credentials for Azure App Service.

Next, you will add your HTML files to the web app. You can use an FTP client such as FileZilla, Visual Studio, Cyberduck, or WinSCP. For more information, see Deploy your app to Azure App Service using FTP/S.

Upload HTML files

1. Open your preferred FTP client. For information about best practices that are related to uploading files to a web app, see Deploy your app to Azure App Service using FTP/S.

2. Enter the host (the FTPS endpoint value from the Deployment Center for the web app), user name, and password, and then connect.

3. Under /site/wwwroot on the host, create a language folder for each language that your custom Help website must support. Upload the HTML files and other associated files to each language folder.

   IMPORTANT
   Remember to use folder names that correspond to the languages that the client expects. For more information, see Language and locale descriptors in the product and in Help.

Your custom Help website has now been deployed to Azure and should be visible in a browser.

Create a storage account

Next, create a storage account that uses a blob container to store the JSON files that the search service will use. For more information about Azure Storage, see the Azure Storage documentation.

You can generate the JSON files from your HTML Help files by using the ConvertHtmlToJson tool that is part of
1. In the Azure portal, go to your resource group, select Add, select Storage account, and specify a name for the storage account, such as mycustomhelpstorage. Then select Review + Create. For more information, see Create a storage account.

2. Validate and create the storage account.

   After the deployment is completed, the new storage account is listed under the resource group.

3. Select your storage account, and then, on the left side of the page, under Blob Service, select Containers. Add a container, and specify a name, such as mycustomhelpcontainer. For more information, see Quickstart: Upload, download, and list blobs with the Azure portal.

You can now upload your JSON files. The folder structure that you use must match the folder structure that you created for the HTML files to match the languages of your solution. For example, if your web app has HTML files in an en-US folder, create an en-US folder in the container, and upload the en-US JSON files to this folder.

There are several ways to upload JSON files to the blob container. If you prefer to use a user interface (UI), Azure Storage Explorer is a convenient tool that lets you use Azure Storage to manage file operations. If you prefer a command-line option, you can use AzCopy. For more information, see Transfer data with the AzCopy on Windows.

Create a search service

Next, create a search service, so that the content can be indexed and is discoverable by the in-product Help pane. For more information, see What is Azure Cognitive Search?

   - In the Azure portal, go to your resource group, select Add, and select Azure Cognitive Search, and then, under URL, specify a name for the service, such as mycustomhelpsearch. Then select Review + Create.

   The service is added to the resource group.

Configure the search service

In the previous section, you created a search service. In this section, you will configure it by creating a data source, an index, and an indexer for each locale. The JSON files that you uploaded to the blob container will then be indexed and searchable. In the examples that follow, the Postman tool is used to make several API calls. However, you can use your own method to call those APIs.

Create a data source

1. Open Postman, and create a new POST request. If you're unfamiliar with this tool, see Explore Azure Cognitive Search REST APIs using Postman.

2. In the Enter request URL field, enter

   https://[AzureSearchServicename].search.windows.net/datasources?api-version=2017-11-11. Replace [AzureSearchServicename] with the name of the search service that you created in the Create a search service section of this topic (for example, mycustomhelpsearch).

3. On the Headers tab, set "Content-type" to application/json, and set api-key to the key from your Azure Cognitive Search service. You can find the key in Access keys under Settings on the left side of the search service.

4. In the Authorization tab, set Type to No Auth.

5. On the Body tab, paste the following text.
Create an index

1. In Postman, create a new POST request that has the following parameters:
   - **URL**: `https://[AzureSearchServicename].search.windows.net/indexes?api-version=2017-11-11` (Replace `[AzureSearchServicename]` with the name of your search service.)
   - **Type** (on the Authorization tab): **No Auth**
   - **Content-Type** (on the Headers tab): **application/json**
   - **api-key** (on the Headers tab): The key from your Azure Cognitive Search service

2. On the **Body** tab, paste the following text.

```json
{
    "name": "[datasourcename]",
    "type": "azureblob",
    "credentials": {
        "connectionString": "DefaultEndpointsProtocol=https;AccountName=
[StorageAccountName];AccountKey=[key];EndpointSuffix=core.windows.net"
    },
    "container": {
        "name": "[JSONStorageContainerName]"
    }
}
```

Replace the following parameters with the relevant values:

- **[datasourcename]** – Specify a name for the data source, such as `mycustomhelpdatasource`.
- **[StorageAccountName]** – Specify the name of the storage account that you created in the Create a storage account section (for example, `mycustomhelpstorage`).
- **[key]** – Specify the access key for your storage account. You can find the key in **Keys** under **Settings** on the left side of the storage account.
- **[JSONStorageContainerName]** – Specify the name of the blob container that you created in the Create a storage account section (for example, `mycustomhelpcontainer`).

6. Select **Send**, and make sure that the value in the **Status** field is **201 Created**.

Next, you will configure the search service so that it has an index of the content for each locale that you want to support.

**Create an index**

1. In Postman, create a new POST request that has the following parameters:

   - **URL**: `https://[AzureSearchServicename].search.windows.net/indexes?api-version=2017-11-11` (Replace `[AzureSearchServicename]` with the name of your search service.)
   - **Type** (on the Authorization tab): **No Auth**
   - **Content-Type** (on the Headers tab): **application/json**
   - **api-key** (on the Headers tab): The key from your Azure Cognitive Search service

2. On the **Body** tab, paste the following text.
Create an indexer

```json
{
    "name": "[IndexName]",
    "fields": [
        { "name": "id", "type": "Edm.String", "key": true, "searchable": true, "filterable": true, "sortable": true, "facetable": true },
        { "name": "title", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true, "facetable": true, "analyzer": "[AnalyzerName]" },
        { "name": "description", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true, "facetable": true, "analyzer": "[AnalyzerName]" },
        { "name": "ms_search_form", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true, "facetable": true },
        { "name": "ms_search_region", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true, "facetable": true },
        { "name": "ms_locale", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true, "facetable": true },
        { "name": "metadata_storage_path", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true, "facetable": true },
        { "name": "metadata_storage_name", "type": "Edm.String", "searchable": true, "filterable": true, "sortable": true, "facetable": true },
        { "name": "metadata_storage_content_type", "type": "Edm.String", "searchable": true, "filterable": false, "sortable": false, "facetable": false }
    ]
}
```

Replace the following parameters with the relevant values:

- **[IndexName]** – Specify the name of the index that should be created, such as *indexenus*.
- **[AnalyzerName]** – Specify the name of the language analyzer that should be used, such as `en.microsoft`.

**NOTE**

The index is language-specific. The *title* and *description* fields contain translations, and it's important that you set the corresponding language analyzer value. Use an appropriate value, based on the language of the index that you're creating. For a list of language analyzers, see *Analyzer List*.

3. Select Send, and make sure that the Status field is set to **201 Created**.

4. If you prepared custom Help content for multiple languages, repeat these steps to create a unique index for each language.

The index isn't an index until it has been processed by an indexer. Think of the table of contents in a reference book. It would not really be useful unless it also lists the page numbers for where to find the various sections in the book. Similarly, the indexer for your search service fills in the index, based on a search. For more information, see *Indexers in Azure Cognitive Search*.

**Create an indexer**

1. In Postman, create a new POST request that has the following parameters:

   - **URL**: `https://[AzureSearchServiceName].search.windows.net/indexers?api-version=2017-11-11` (Replace [AzureSearchServiceName] with the name of your search service.)
   - **Type** (on the Authorization tab): *No Auth*
   - **Content-Type** (on the Headers tab): application/json
   - **api-key** (on the Headers tab): The key from your Azure Cognitive Search service

2. On theBody tab, paste the following text.
Replace the following parameters with the relevant values:

- **[IndexerName]** – Specify the name of the indexer that should be created, such as `indexerenus`.
- **[DatasourceName]** – Specify the name of the data source that you created, such as `mycustomhelpdatasource`.
- **[IndexName]** – Specify the name of the index that you created, such as `indexenus`.

**NOTE**
This configuration will set up an indexer that is scheduled to run every 10 hours ("schedule": { "interval": "PT10H" }). However, you can adjust the interval as appropriate.

3. Select **Send**, and make sure that the **Status** field is set to **201 Created**.

4. If you prepared custom Help content for multiple languages, repeat these steps to create a unique indexer for each index.

**NOTE**
An index will contain data only after its indexer has run. You might want to manually run the indexer if you're planning to test the index immediately.

**IMPORTANT**
If you update your content, remember to regenerate the JSON files and upload them to the storage service. If you add content, you can either manually run the indexers or wait for the next scheduled run. Your updated content won't be available in search results until the next time that the indexers are run.

You can optionally use Postman to test the search. For an example that shows how to use Postman for testing, see **Search your JSON files**.

**Next steps**

If you completed all the steps in this topic, your Help content has now been uploaded to an Azure web app, and it has been indexed.

The next step is to extend the **Help** pane so that it can detect your content. In that way, when users open the **Help** pane in your Dynamics 365 solution, the in-product **Help** pane will be able to generate context-sensitive links to your Help. For more information, see **Connect a custom Help website to the Help pane**.
See also

- Custom Help overview
- Custom Help Toolkit
- Language and locale descriptors in the product and in Help
Connect a custom Help website to the Help pane

If you deliver custom Help content for a Finance and Operations solution, you can extend the Help pane so that it consumes that content. You complete this one-time configuration by using the Finance and Operations development environment in Microsoft Visual Studio. After you’ve finished, users can select among tabs for task guides, Microsoft Help content, and your Help content.

The process for connecting your custom Help website to the in-product Help pane involves the following steps:

1. Extend the Help pane in Visual Studio.
2. Assign an index to a language.
3. Customize language fallback.

**IMPORTANT**
The procedures that follow require the development tools for Finance and Operations apps in Visual Studio. For more information, see Development tools in Visual Studio.

Extend the Help pane and assign the custom Help indexes to languages

The Help Pane extension folder of the Custom Help Toolkit contains the AzureSearchCustomHelp solution that you can open in the Finance and Operations development environment. That folder also contains the HelppaneOption.axpp project that you can then import into the solution in Visual Studio.

**Extend the Help pane**

1. In the Finance and Operations development environment, open the AzureSearchCustomHelp.sln solution.
2. On the Dynamics 365 menu, select Import project.
3. In the File name field, specify the path of the HelppaneOption.axpp project, and then select OK to complete the import process. Update the references so that no references are missing.
4. In the HelppaneMacro file, update the values of the following parameters:

   - ![WebAppName](#) – Specify the name of the web app that you created in Create a web app. For example, specify MyCustomHelpWebApp.
   - ![Admin key value](#) – Specify the admin key for the Azure Cognitive Search service. You can find the key in Access keys under Settings on the left of the search service in the Azure portal.
   - ![SearchServiceName](#) – Specify the name of the search service that you created in Create a search service. For example, specify mycustomhelpsearch.

The following example shows the content of the HelppaneMacro file.
#define.webApp('http://[WebAppName].azurewebsites.net/')
#define.queryApiKey('Admin key value')
#define.defaultstring('dashboard')
#define.searchservicename('[SearchServiceName]')
#define.CustomResultError('error')
#define.CustomTabPage('CustomTabPage')
#define.CustomHelp('Custom Help')
#define.CustomTitle('CustomTitle')
#define.htm('html')

5. Optional: If you want to change any of the user interface (UI) strings that appear in the Help pane, edit
the Customhelppane.en-US.label.txt file.
Next, you must specify the language that the search index for your custom Help is intended for.
Assign a custom index to a language
1. Open the Language.config file in the solution.
2. In the list, find the language of the index, and specify an index name by using the index="" ,
parentindex="" , or ultimateindex="" key.
For example, you created search indexes for English (United States) and German (Austria), and you
named them myenusindex and mydeatindex , respectively. In this case, here is what your entries will
look like.
<add language="en-US" ultimateindex="myenusindex" />
<add language="de-AT" parentlanguage="de" index="mydeatindex" />

3. Optional: Customize language fallback for your index, as described in the next section.
4. Build the AzureSearchCustomHelp solution.
The result is a model that you upload to the Asset library of the customer project or the solution project in
Microsoft Dynamics Lifecycle Services (LCS).

Customize language fallback
Language fallback means that the Help pane runs a search in additional languages if the intended language
either doesn't return a result or doesn't exist.
NOTE
A custom index must be available for the additional languages.

The search and fallback order have the following order of priority:
1. The language that is set in the client (for example, de-AT )
2. The language that is defined in the parentlanguage attribute for that language (for example, <add
language="de-AT" parentlanguage="de" index="mydeindex" /> )
3. The language that the ultimateindex attribute is set for (for example, <add language="en-US"
ultimateindex="myenusindex" /> )
IMPORTANT
If the parentlanguage attribute is set, there must be a corresponding parentindex key.


The following scenario is valid, because language="de" has parentindex="indexde", and both de-DE and de-AT are descendants of de.

```
<add language="de" parentindex="indexde"/>
<add language="de-DE" parentlanguage="de" index=""/>
<add language="de-AT" parentlanguage="de-DE" index="indexdeat"/>
```

For more information about languages, see Languages, translations, and adaptations.

The following sections provide sample configurations.

### Help content for one locale

In this configuration, you have Help content only for English (United States). Regardless of the locale that clients are set to, they will show the Help content in English (United States).

```
<add language="en-US" ultimateindex="indexenus"/>
```

### Help content for multiple locales

In this configuration, you have Help content for French, German, and English (United States). Clients that are set to the de locale will show the Help content in German, clients that are set to the fr locale will show the content in French, and clients that are set to any other locale will show the content in English (United States).

```
<add language="en-US" ultimateindex="indexenus"/>
<add language="fr" parentindex="indexfr"/>
<add language="de" parentindex="indexde"/>
```

If clients are set to the de or fr locale, but no results are found in the German or French content, respectively, results will be shown in English (United States), if content is available in that language.

### Help content that uses parent locales

In this configuration, you have Help content for German, German (Austria), and English (United States). For example, you have several topics that are related specifically to features for Austria, but topics in German can be used otherwise.

```
<add language="en-US" ultimateindex="indexenus"/>
<add language="de" parentindex="indexde"/>
<add language="de-AT" parentlanguage="de" index="indexdeat"/>
```

If the client is set to the de-AT locale, but no results are found in the German (Austria) content, results will be shown in German and English (United States), if content is available in those languages.

### See also

- [Deploy custom Help to Azure](#)
- [Custom Help Toolkit](#)
- [Language and locale descriptors in the product and in Help](#)
- [Configure the Help experience for Finance and Operations apps](#)
- [Help system](#)
Microsoft has published a GitHub repository (repo) that includes scripts and tools that can help you prepare context-sensitive Help for customized Finance and Operations solutions. This context-sensitive Help can be accessed from the in-product Help pane.

Tools in the toolkit

The Custom Help Toolkit is available at https://github.com/microsoft/dynamics365f-o-custom-help. The repo contains the following tools and the source code for those tools:

- The **HtmlFromRepoGenerator** tool
  For more information, see Custom Help Toolkit: The HtmlFromRepoGenerator tool.

- The **ConvertHtmlToJson** tool
  For more information, see Custom Help Toolkit: The ConvertHtmlToJson tool.

- The **HtmlLocaleChanger** tool
  For more information, see Custom Help Toolkit: The HtmlLocaleChanger tool.

- The **Help Pane extension** Microsoft Visual Studio project
  For more information, see Connect a custom Help website to the Help pane.

- Dynamics AX 2012 metadata scripts
  For more information, see Convert Dynamics AX custom Help for use in Dynamics 365.

**NOTE**

The first version of the toolkit is available as a release in the GitHub repo.

See also

- Custom Help overview
- Deploy custom Help to Azure
- Connect a custom Help website to the Help pane
- Language and locale descriptors in the product and in Help
- Convert Dynamics AX custom Help for use in Dynamics 365
The Custom Help Toolkit includes the `HtmlFromRepoGenerator` tool, which gets Microsoft content in Markdown files and converts it to HTML files. You can then deploy the HTML files to a website.

### Use the HtmlFromRepoGenerator tool to get Markdown files and generate HTML files

The `HtmlFromRepoGenerator` tool provides functionality that supports the creation of custom Help that is based on source files from Microsoft. You can use the tool to perform these tasks:

- Clone a Microsoft documentation repository (repo).
- Remove developer and admin content from your clone of the Microsoft repo.
- Update links to files that are no longer present in the clone.
- Update the value of the `ms.locale` property so that it matches the language options that are supported by the Finance and Operations client.

  The language descriptors that the client uses differ from the language descriptors that are used in the corresponding GitHub repos. Before localized custom Help can be called, the language descriptors in the source content must be changed so that they match the client's languages. For more information, see [Language and locale descriptors in the product and in Help](#).

- Generate HTML files that can be used to publish content.

  The HTML files will be generated in the `d365F-O` subfolder. The files are generated based on style sheets and templates that are part of the tool. For more information, see the [Modifying the styling of the generated HTML files](#) section of this topic.

- Compare a localized Microsoft repo to the en-US repo to identify differences and update links accordingly.

### Syntax

Here is the syntax for running `HtmlFromRepoGenerator.exe`.

```
```

Here is an explanation of the parameters.
<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Json</td>
<td>Specify a relative path for the location of the <code>docfx.json</code> file. In Microsoft documentation repos, this location is typically <code>articles/</code>.</td>
</tr>
<tr>
<td>Out</td>
<td>Specify the folder where your existing cloned repo exists, or the folder to clone the repo to. If you run the <code>HtmlFromRepoGenerator</code> tool to clone a repo, this folder must not already exist. Use the language name as the folder name, as described in <a href="#">Language and locale descriptors in the product and in Help</a>.</td>
</tr>
<tr>
<td>ExternalText</td>
<td>Specify text that should be added to the updated links if the <code>HtmlFromRepoGenerator</code> tool must replace the original links.</td>
</tr>
<tr>
<td>DoNotClone</td>
<td>Set this parameter when you run the tool against a previously cloned repo.</td>
</tr>
</tbody>
</table>
| Repo      | Specify the repo URL. This parameter is optional if you run the tool against a previously cloned repo. Examples of URLs for Microsoft documentation repos include:

- [https://github.com/MicrosoftDocs/Dynamics-365-Unified-Operations-public](https://github.com/MicrosoftDocs/Dynamics-365-Unified-Operations-public) for English (United States) and
- [https://github.com/MicrosoftDocs/Dynamics-365-Operations.de-de](https://github.com/MicrosoftDocs/Dynamics-365-Operations.de-de) for German (Germany). |
| RemoveGitFolder | Specify whether the `.git` folder should be removed. |
| ReplaceUrl | Specify the URL that should replace links between files when the target files aren't present. This parameter is intended to be used to turn relative links into absolute links. |
| LogsDir   | Specify the folder to save logs files to. |

The following additional parameters are used when the tool is run against the localized Microsoft documentation repos.

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnRepo</td>
<td>Specify the URL of the en-US repo. This parameter is optional if you run the tool against a previously cloned repo. The URL of the Microsoft documentation repo for English (United States) is <a href="https://github.com/MicrosoftDocs/Dynamics-365-Unified-Operations-public">https://github.com/MicrosoftDocs/Dynamics-365-Unified-Operations-public</a>.</td>
</tr>
<tr>
<td>EnOut</td>
<td>Specify the folder where the en-US repo exists, or the folder to clone it to. If you run the tool against a previously cloned repo, this folder must not already exist.</td>
</tr>
</tbody>
</table>
**PARAMETER** | **DESCRIPTION**  
--- | ---  
Lng | Specify the language value that should be used for `ms.locale` metadata in the generated HTML files. The value must correspond to the value that is specified in the language settings of the Finance and Operations client. If this parameter isn’t set, the tool uses `en-US`. For more information, see [Language and locale descriptors in the product and in Help](#).  
Rtl | Include this parameter if the language uses right-to-left (RTL) formatting. Examples of RTL languages include Arabic and Hebrew.  

### Examples

**NOTE**  
Because the Microsoft repos contain many files, the process takes several minutes. If you run the tool against multiple localized repos, the process takes longer.

The following example clones the en-US repo and generates HTML files for en-US.

```bash  
```

The following example uses a previously cloned en-US repo and generates HTML files for en-US.

```bash  
HtmlFromRepoGenerator.exe --json articles/ --out "D:\D365-Operations\en-US" --externalText "(This is an external link)" --replaceUrl "https://docs.microsoft.com/en-us/dynamics365/supply-chain" --LogsDir D:\D365-Operations\logs\en-US  
```

The following example clones the de-DE and en-US repos, and generates HTML files for de.

```bash  
HtmlFromRepoGenerator.exe --json articles/ --out "D:\D365-Operations\de" --repo "https://github.com/MicrosoftDocs/Dynamics-365-Operations.de-de" --externalText "(This is an external link)" --EnRepo "https://github.com/MicrosoftDocs/Dynamics-365-unified-Operations-public" --EnOut "D:\D365-Operations\en-us" --replaceUrl "https://docs.microsoft.com/de-de/dynamics365/supply-chain" --lng "de" --LogsDir D:\D365-Operations\logs\de  
```

The following example uses the existing de-DE and en-US repos, and generates HTML files for de. If you use the existing de-DE repo, make sure that it’s up to date.

```bash  
HtmlFromRepoGenerator.exe --json articles/ --out "D:\D365-Operations\de" --DoNotClone --externalText "(This is an external link)" --enOut "D:\D365-Operations\en-us" --replaceUrl "https://docs.microsoft.com/de-de/dynamics365/supply-chain" --lng "de" --LogsDir D:\D365-Operations\logs\de  
```
Modifying the styling of the generated HTML files

The HTML files that the **HtmlFromRepoGenerator** tool generates are based on a set of predefined templates. In most cases, you can edit the style sheets in the `d365F-O\styles` folder to change the appearance of your content.

For advanced scenarios, you can change the templates that the **HtmlFromRepoGenerator** tool uses. The source files are included in the `SourceCode` folder in the GitHub repo. The templates are in the `SourceCode\HtmlFromRepoGenerator\HtmlFromRepoGenerator\HtmlFromRepoGenerator\Resources` subfolder.

If you change the templates, you must rebuild `HtmlFromRepoGenerator.exe`.

For more information, see [*Introduction to DocFX Template System*](#).

See also

- Custom Help Toolkit
- Custom Help overview
- Deploy custom Help to Azure
- Language and locale descriptors in the product and in Help
- Convert Dynamics AX custom Help for use in Dynamics 365
The Custom Help Toolkit includes the ConvertHtmlToJson tool, which converts HTML files to JavaScript Object Notation (JSON) files. The search service uses the JSON files to index Help content.

Use the ConvertHtmlToJson tool to generate JSON files

The ConvertHtmlToJson tool transforms HTML files into JSON files. You can then add the JSON files to the Microsoft Azure Search service, which will generate context-sensitive links to your Help content.

The JSON files include metadata that the indexer uses to identify the form and language that the target Help page is intended for.

Here is the syntax for running ConvertHtmlToJson.exe.

```
ConvertHtmlToJson.exe --h <path> -j <path> --v <true|false>
```

Here is an explanation of the parameters.

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>h</td>
<td>Specify the path of the HTML files to process.</td>
</tr>
<tr>
<td>j</td>
<td>Specify the folder to save the JSON files to. The specified folder must already exist.</td>
</tr>
<tr>
<td>v</td>
<td>Set this parameter to true to turn on verbose logging. Otherwise, set it to false.</td>
</tr>
</tbody>
</table>

Example

The following example generates JSON files without verbose logging.

```
ConvertHtmlToJson.exe --h D:\D365-Operations\d365F-O\supply-chain\de -j D:\D365-Operations\json\supply-chain\de
```

See also

Custom Help overview
Deploy custom Help to Azure
Language and locale descriptors in the product and in Help
Convert Dynamics AX custom Help for use in Dynamics 365
The Custom Help Toolkit includes the 
**HtmlLocaleChanger** tool, which can process HTML files that are generated by the 
**HtmlFromRepoGenerator** tool.

### Use the HtmlLocaleChanger tool to align locales

The **HtmlLocaleChanger** tool can update your HTML files by setting a new value for the **ms.locale** property. For example, you have HTML files for German (Germany), and you want to make the same content available in German (Austria). In this case, you can run the tool to change the setting from **ms.locale: de-de** to **ms.locale:de-at**.

Here is the syntax for running HtmlLocaleChanger.exe.

```
HtmlLocaleChanger.exe --h <path> --l <locale> --v <true|false>
```

Here is an explanation of the parameters.

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>h</td>
<td>Specify the path of the HTML files to process.</td>
</tr>
<tr>
<td>l</td>
<td>Specify the new locale for the HTML files.</td>
</tr>
<tr>
<td>v</td>
<td>Set this parameter to <strong>true</strong> to turn on verbose logging. Otherwise, set it to <strong>false</strong>.</td>
</tr>
</tbody>
</table>

### Example

The following example changes the locale to **de-at** and turns on verbose logging.

```
HtmlLocaleChanger.exe --h D:\D365-Operations\d365F-0\supply-chain\de --l de-at --v
```

### See also

- Custom Help overview
- Deploy custom Help to Azure
- Language and locale descriptors in the product and in Help
- Convert Dynamics AX custom Help for use in Dynamics 365
If you have existing content from Microsoft Dynamics AX 2012, you can reuse it for Dynamics 365 Finance, Dynamics 365 Supply Chain Management, and Dynamics 365 Commerce. However, you must first transform the HTML files so that they can be used in the custom Help environment.

Converting AX 2012 content

The Microsoft Custom Help Toolkit includes a Windows PowerShell script, `run_ax2012.ps1`, that can transform AX 2012 HTML files so that they can be used in the custom Help environment. The script makes the following changes to the AX 2012 HTML files:

- Replace the `Microsoft.Help.F1` metadata name with `ms.search.form`.
- Replace the `Title` metadata name with `title`.
- Change the file name extension from `.htm` to `.html`.
- Add the following metadata.

```xml
<meta name="ms.search.region" content="Global" />
<meta name="ms.search.scope" content="Operations, Core" />
<meta name="ms.dyn365.ops.version" content="AX 7.0.0" />
<meta name="ms.search.validFrom" content="2016-05-31" />
<meta name="ms.search.industry" content="cross" />
```

Running the script

You can run the following command from a Command Prompt window, or you can run the script directly in Windows PowerShell.

```
PowerShell.exe -File run_ax2012.ps1 "Original file location" "New file location"
```

The following metadata is currently used, or it's reserved so that it can be used during indexing.

```xml
<meta name="title" content="Title of file" />
<meta name="ms.locale" content="locale" />
<meta name="ms.search.form" content="FormAOTName" />
<meta name="description" content="Description of file" />
<meta name="ms.search.region" content="Global" />
<meta name="ms.search.scope" content="Operations, Core" />
<meta name="ms.dyn365.ops.version" content="AX 7.0.0" />
<meta name="ms.search.validFrom" content="2016-05-31" />
<meta name="ms.search.industry" content="cross" />
```

For a description of the required metadata, see Metadata requirements for custom Help topics.

Moving to Markdown

To convert your existing content to Markdown, you can use various third-party tools, such as PanDoc and the
**Writage** plug-in for Word.

After you've converted your content to Markdown, you can use open-source tools such as **DocFx** to generate content for your website. In general, by working in Markdown, you gain access to a wide range of open-source tools. For more information, see **Extend, customize, and collaborate on the Help**.

**See also**

- Custom Help overview
- Custom Help Toolkit
- Extend, customize, and collaborate on the Help
The client that Finance and Operations apps use supports multiple languages and locales. To add custom Help content for one or more locales to the in-product Help pane, you must make sure that both the following conditions are met:

- The value of the `ms.locale` property in each HTML file matches the locale of the content.
  
  For example, the German (Germany) content must have a setting of `ms.locale: de-de`.

- On the custom Help website, the content is in a folder that has the same name as the locale.
  
  For example, the German (Germany) content must be in a folder that is named `de-de`.

For more information, see Custom Help overview and Deploy custom Help to Azure.

**Languages and descriptors**

The following table maps the language names between the Finance and Operations client and the GitHub repositories (repos) that contain translated Microsoft Help content.

<table>
<thead>
<tr>
<th>LANGUAGE/LOCALE IN THE CLIENT</th>
<th>LANGUAGE/REGION NAME</th>
<th>NAME OF THE GITHUB REPO</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
<td>Arabic (Saudi Arabia)</td>
<td>Dynamics-365-Operations.ar-sa</td>
</tr>
<tr>
<td>ar-ae</td>
<td>Arabic (United Arab Emirates)</td>
<td>Dynamics-365-Operations.ar-ae</td>
</tr>
<tr>
<td>cs</td>
<td>Czech</td>
<td>Dynamics-365-Operations.cs-cz</td>
</tr>
<tr>
<td>da</td>
<td>Danish</td>
<td>Dynamics-365-Operations.da-dk</td>
</tr>
<tr>
<td>de</td>
<td>German (Germany)</td>
<td>Dynamics-365-Operations.de-de</td>
</tr>
<tr>
<td>de-at</td>
<td>German (Austria)</td>
<td>Dynamics-365-Operations.de-de</td>
</tr>
<tr>
<td>de-ch</td>
<td>German (Switzerland)</td>
<td>Dynamics-365-Operations.de-de</td>
</tr>
<tr>
<td>en-au</td>
<td>English (Australia)</td>
<td>Dynamics-365-Unified-Operations-Public</td>
</tr>
<tr>
<td>en-ca</td>
<td>English (Canada)</td>
<td>Dynamics-365-Unified-Operations-Public</td>
</tr>
<tr>
<td>en-gb</td>
<td>English (United Kingdom)</td>
<td>Dynamics-365-Unified-Operations-Public</td>
</tr>
<tr>
<td>en-ie</td>
<td>English (Ireland)</td>
<td>Dynamics-365-Unified-Operations-Public</td>
</tr>
<tr>
<td>LANGUAGE/LOCALE IN THE CLIENT</td>
<td>LANGUAGE/REGION NAME</td>
<td>NAME OF THE GITHUB REPO</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>en-in</td>
<td>English (India)</td>
<td>Dynamics-365-Unified-Operations-Public</td>
</tr>
<tr>
<td>en-my</td>
<td>English (Malaysia)</td>
<td>Dynamics-365-Unified-Operations-Public</td>
</tr>
<tr>
<td>en-nz</td>
<td>English (New Zealand)</td>
<td>Dynamics-365-Unified-Operations-Public</td>
</tr>
<tr>
<td>en-sg</td>
<td>English (Singapore)</td>
<td>Dynamics-365-Unified-Operations-Public</td>
</tr>
<tr>
<td>en-us</td>
<td>English (US)</td>
<td>Dynamics-365-Unified-Operations-Public</td>
</tr>
<tr>
<td>en-za</td>
<td>English (South Africa)</td>
<td>Dynamics-365-Unified-Operations-Public</td>
</tr>
<tr>
<td>es</td>
<td>Spanish (Spain)</td>
<td>Dynamics-365-Operations.es-es</td>
</tr>
<tr>
<td>es-mx</td>
<td>Spanish (Mexico)</td>
<td>Dynamics-365-Operations.es-es</td>
</tr>
<tr>
<td>et</td>
<td>Estonian</td>
<td>Dynamics-365-Operations.et-ee</td>
</tr>
<tr>
<td>fi</td>
<td>Finnish</td>
<td>Dynamics-365-Operations.fi-fi</td>
</tr>
<tr>
<td>fr</td>
<td>French (France)</td>
<td>Dynamics-365-Operations.fr-fr</td>
</tr>
<tr>
<td>fr-be</td>
<td>French (Belgium)</td>
<td>Dynamics-365-Operations.fr-fr</td>
</tr>
<tr>
<td>fr-ca</td>
<td>French (Canada)</td>
<td>Dynamics-365-Operations.fr-fr</td>
</tr>
<tr>
<td>fr-ch</td>
<td>French (Switzerland)</td>
<td>Dynamics-365-Operations.fr-fr</td>
</tr>
<tr>
<td>hu</td>
<td>Hungarian</td>
<td>Dynamics-365-Operations.hu-hu</td>
</tr>
<tr>
<td>is</td>
<td>Icelandic</td>
<td>Dynamics-365-Operations.is-is</td>
</tr>
<tr>
<td>it</td>
<td>Italian</td>
<td>Dynamics-365-Operations.it-it</td>
</tr>
<tr>
<td>it-ch</td>
<td>Italian (Switzerland)</td>
<td>Dynamics-365-Operations.it-it</td>
</tr>
<tr>
<td>ja</td>
<td>Japanese</td>
<td>Dynamics-365-Operations.ja-jp</td>
</tr>
<tr>
<td>lt</td>
<td>Lithuanian</td>
<td>Dynamics-365-Operations.lt-lt</td>
</tr>
<tr>
<td>lv</td>
<td>Latvian</td>
<td>Dynamics-365-Operations.lv-lv</td>
</tr>
<tr>
<td>nb-no</td>
<td>Norwegian Bokmål</td>
<td>Dynamics-365-Operations.nb-no</td>
</tr>
<tr>
<td>nl</td>
<td>Dutch (Netherlands)</td>
<td>Dynamics-365-Operations.nl-nl</td>
</tr>
<tr>
<td>LANGUAGE/LOCALE IN THE CLIENT</td>
<td>LANGUAGE/REGION NAME</td>
<td>NAME OF THE GITHUB REPO</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>nl-be</td>
<td>Dutch (Belgium)</td>
<td>Dynamics-365-Operations.nl-nl</td>
</tr>
<tr>
<td>pl</td>
<td>Polish</td>
<td>Dynamics-365-Operations.pl-pl</td>
</tr>
<tr>
<td>pt-br</td>
<td>Portuguese (Brazil)</td>
<td>Dynamics-365-Operations.pt-br</td>
</tr>
<tr>
<td>ru</td>
<td>Russian</td>
<td>Dynamics-365-Operations.ru-ru</td>
</tr>
<tr>
<td>sv</td>
<td>Swedish</td>
<td>Dynamics-365-Operations_sv-se</td>
</tr>
<tr>
<td>th</td>
<td>Thai</td>
<td>Dynamics-365-Operations.th-th</td>
</tr>
<tr>
<td>tr</td>
<td>Turkish</td>
<td>Dynamics-365-Operations.tr-tr</td>
</tr>
<tr>
<td>zh-hans</td>
<td>Chinese</td>
<td>Dynamics-365-Operations.zh-cn</td>
</tr>
</tbody>
</table>

Languages, translations, and adaptations

Microsoft teams create content in English (United States). That content is then translated into several languages, and the translated content is made available in a public GitHub repo for each language.

To maximize reuse of translations, translation services treat some languages as variants of another language. For example, German for Austria and German for Germany are considered so closely related that they are treated as variants of each other. Therefore, you can use the files in the Dynamics-365-Operations.de-de GitHub repo as a starting point for both German (Germany) content and German (Austria) content.

When you extend the in-product Help pane, you must assign indexes for the relevant locales. For more information, see Customize language fallback.

See also

Custom Help overview
Custom Help Toolkit
The source files for the Microsoft Help for Finance and Operations apps are available in public GitHub repositories (repos). Any solution provider can easily extend and customize the content for specific solutions. This topic explains how to work with the GitHub repos and Markdown files.

For information about how to create Markdown files in GitHub repos, see the Docs contributor guide. For information about how to deploy custom Help, see Custom Help overview.

Contribute to the content

One benefit of GitHub is that you can contribute to the core content that the Microsoft team provides in the MicrosoftDocs/Dynamics-365-Unified-Operations-public repo. For example, you have a new topic that you think will be helpful to other users, or you have a correction to an existing topic. If you want to contribute to the Dynamics-365-Unified-Operations-public repo, you can create a pull request from your repo to the Dynamics-365-Unified-Operations-public repo. The Microsoft team will then review the request and include your changes as appropriate.

You can also contribute and make edits to the existing documentation. To get started, select the Edit button (pencil symbol) in a topic. The following video shows how you can contribute to the Microsoft documentation.

NOTE

Microsoft currently accepts pull requests only to the Dynamics-365-Unified-Operations-public repo, not to the language-specific repos. If you have feedback about translations, you can report a GitHub issue in the relevant repo.

Extend and customize Microsoft source content from GitHub repos

Microsoft uses separate repos in GitHub for the source content and for each language that Microsoft translates content into. The Dynamics-365-Unified-Operations-public repo contains the source content in English (United States). If you want to access the content in other languages, the names follow the pattern Dynamics-365-Operations.<language>-<country>. For example, the version for German (Germany) is named Dynamics-365-Operations.de-de.

When Microsoft publishes an update to the content, the main branch in the corresponding GitHub repo is updated. The source repo is updated weekly. However, the related language-specific repos are updated less often. The frequency depends on when new translations are made available. If you fork one of the Microsoft repos, you can choose to update your fork with updates from the Microsoft repo on a monthly basis or less often, depending on your preferred work processes. The GitHub platform and tooling will help you manage any potential merge conflicts if you change files that Microsoft has also changed. For more information, see Set up Git repository locally for documentation in the Docs authoring guide and Fork a repo in the Help for GitHub.

IMPORTANT

In April 2021, the default branch in the public source repo has been renamed from live to main. If you have any scripts that rely on the live branch, please update them to rely on main instead. The default branches in the language-specific repos will be renamed later.
Get started with GitHub

To join Microsoft in the world of GitHub and Markdown, you must be familiar with some new terminology and tools. The following list outlines the main steps, but you can find additional content, tools, and ideas in the GitHub documentation and other forums.

1. Sign up for GitHub.

   For more information, see GitHub account setup and Install content authoring tools in the Docs contributor guide.

2. Fork the appropriate repo.

   To extend and customize Microsoft content for a custom Help solution, you must create a fork of the repo. If you want to customize Microsoft content in Markdown format, we recommend that you manually fork the relevant repo and use your favorite Markdown editor. For more information, see Set up Git repository locally for documentation and Git and GitHub essentials for Docs in the Docs contributor guide.

   TIP
   You aren't required to make your GitHub repos public. When you fork a public repo, in the settings for the new repo, you can specify whether the repo is public, private, or available only to specific GitHub accounts.

Markdown format

The syntax that is used to format text for topics is named Markdig Flavored Markdown. This syntax complies with CommonMark. To learn more about how to work with Markdown, see Getting started with writing and formatting on GitHub.

You can convert content from Microsoft Word to Markdown by using open-source tools or other tools. In this way, you can easily recycle content.

Get updates from Microsoft

Microsoft makes frequent changes to the content, and those changes show up in the public GitHub repos. The base repo, MicrosoftDocs/Dynamics-365-Unified-Operations-public, is updated weekly. However, you can choose to get updates monthly, twice a year, or once a year, for example. The translation repos are updated less frequently, so you might want a monthly schedule or less frequent updates, as appropriate.

When you decide that it’s time to get the latest version of the content from Microsoft, you can use the Git command line or GitHub Desktop. The Help for GitHub provides an example that shows how this process works in GitBash. In GitHub Desktop, you use the Merge into current branch command to pull changes from the origin into your fork.

If your solution is available in more than one country or region, you will probably want to make the content available in multiple languages. Although Microsoft has a GitHub repo for each supported language, the configuration files are available only in the English (United States) version of the base repo, MicrosoftDocs/Dynamics-365-Unified-Operations-public. You can use the HtmlFromRepoGenerator tool from the Custom Help Toolkit to get the files.

Because the Microsoft repos are public, you don’t have to have a valid GitHub account to get the content. However, we recommend that, at a minimum, your organization have a system account that has access to
GitHub.

For more information, see Custom Help Toolkit.

Get the content without a GitHub account

If you don’t want to collaborate with Microsoft on the content, you can get the latest version of the content from GitHub even if you don’t have a GitHub account. For example, you can just clone the relevant GitHub repo. A GitHub account isn’t required to clone a repo. Because the Microsoft repos are public, anyone can always access them.

Translate the content

You can use the Dynamics 365 Translation Service (DTS) to translate your own or the Microsoft-provided content into other languages. The service is hosted in Microsoft Dynamics Lifecycle Services (LCS), and currently supports translation of content in Word documents and HTML files. For more information, see Translate documentation files.

See also

Convert Dynamics AX custom Help for use in Dynamics 365
Docs contributor guide
Docs Authoring Pack for Visual Studio Code
Getting started with writing and formatting on GitHub
Visual Studio Code
Atom
DocFx
The Microsoft global network of datacenters offers the following localization options to help you meet data residency, sovereignty, and compliance requirements:

- **Dynamics 365 Finance and Dynamics 365 Supply Chain Management in US Government Community Cloud (GCC)**
- **Finance and Operations apps in France**
- **Dynamics 365 Finance and Dynamics 365 Supply Chain Management operated by 21Vianet in China**

**See also**

For information about product availability per country or region and workload, see [Dynamics 365 and Power Platform availability](#).
Select Microsoft Dynamics 365 United States (US) Government products are available to qualified government and private entities. Those entities are limited to the following types:

- US federal, state, local, tribal, and territorial government entities
- Private entities that use Dynamics 365 US Government to provide solutions to government entities or to qualified members of the cloud community
- Private entities that have customer data that is subject to government regulations, and Dynamics 365 US Government is the appropriate service to meet the regulatory requirements

For information, see Dynamics 365 US Government.

**Onboarding**

To complete the initial onboarding for an implementation project in Microsoft Dynamics Lifecycle Services (LCS), follow the instructions in Onboard an implementation project. However, don't use the link to public LCS that is provided in those instructions. Instead, use the following URL to open LCS for US Government Community Cloud (GCC): https://gov.lcs.microsoftdynamics.us.

After the initial onboarding is completed, follow the instructions in Project onboarding. Once again, use LCS for GCC instead of public LCS.

**Environment deployment**

After you've completed project onboarding, you can review the additional capabilities of LCS that are described in Lifecycle Services (LCS) for Finance and Operations apps customers. Then move on to environment deployment.

- To deploy Microsoft-managed environments via LCS, follow the instructions in Lifecycle Services (LCS) for Finance and Operations apps customers.
- For cloud-hosted environments, see Deploy and access development environments. You must also complete the Resource Manager onboarding process for your connectors, as described in Complete the Azure Resource Manager onboarding process for US government Lifecycle Services projects.

**NOTE**

For US Government LCS projects, only Azure Government-specific Azure subscriptions are supported.

**Features that aren't available**

Some features won't be available for deployment in GCC, or they won't be available to use with Dynamics 365 in GCC. For example, Azure DevOps Services will be unavailable in GCC. However, on-premises Azure DevOps or public Azure DevOps services can be used. For detailed information about feature availability, see Business Applications US Government Feature Availability.
Frequently asked questions

Are Dynamics 365 Finance and Dynamics 365 Supply Chain Management supported in GCC-High?
No. Dynamics 365 Finance and Dynamics 365 Supply Chain Management are supported only in GCC.

Can I use public Azure DevOps with Finance and Supply Chain Management in GCC?
Yes, you can use public Azure DevOps services if you don't have requirements for a solution that is certified by the Federal Risk and Authorization Management Program (FEDRAMP). Alternatively, you can use Azure DevOps Server.

Can I deploy a cloud-hosted environment Tier-1 development environment on an Azure commercial subscription?
No. In LCS for GCC, you must use an Azure Government subscription to deploy a cloud-hosted environment.

What can I do if I need a package from the Shared asset library, but it isn't available in LCS for GCC?
You can download the same package from the Shared asset library in public LCS. Alternatively, your partner can help you download the package.

Is the code upgrade tool available in GCC?
No, the code upgrade tool isn't currently available in GCC. However, you can create a prospect project in public LCS and use the code upgrade tool. Note that you won't be able to deploy environments in prospect projects.

Can my partner open a support ticket on my behalf?
Yes. However, if your partner uses a non-GCC identity, the support ticket will be directed to the public support queue. We recommend that you use customer GCC entitlement in LCS to open support tickets.

See also

- Dynamics 365 US Government
- Business Applications US Government Feature Availability.
- Lifecycle Services (LCS) user guide
- Cloud deployment overview
Microsoft Dynamics 365 online services operated by 21Vianet is designed to comply with regulatory requirements in China. The services are a physically separated instance of cloud services operated and transacted by a local operator, Shanghai Blue Cloud Technology Co., Ltd (“21Vianet”). This is a wholly owned subsidiary of Beijing 21Vianet Broadband Data Center Co., Ltd. located in mainland China.

Microsoft strives to maintain functional parity between our commercially available service and Finance and Operations apps operated by 21Vianet in China. However, there are notable exceptions to this, which are affected by dependent service or partner-solution availability, market priorities, or compliance regulations.

Provisioning

Customers in China have two options from which to select how they want to access Finance and Supply Chain Management apps.

- Services operated by 21Vianet in China - 21Vianet operates and offers Finance and Supply Chain Management services in China. This option provides a consistent application experience that is the same as global offerings. This option also meets the demands of customers who prefer to use online services provided by a local company that stores their data within China. These services are subject to Chinese laws.

- Services operated by Microsoft – This option is for Finance and Supply Chain Management customers that prefer to use services managed and delivered by Microsoft. For all new customers and existing customers, if the customer purchases Microsoft Azure, Dynamics 365, and Office using an Enterprise Agreement, Microsoft 365 and/or Dynamics 365 can co-exist on the tenant.

For information on provisioning environments, see Create and manage environments in the Power Platform Admin center.

Features not available

Due to certain technical dependencies, the following features listed will not be available for general availability of the Dynamics 365 Services operated by 21Vianet. For information about future feature availability, see Business applications and platform release plans.

- **Development, build, and testing of customizations** will be unavailable in Azure DevOps in Mainland China. However, use of Azure DevOps on-premises will be available in China in April 2019. Also, Azure DevOps can be used in other regions. For more information, see Developer guide for Azure China 21Vianet.

- **Set up and maintain vendor collaboration** will be unavailable due to Azure Active Directory limitations.

- **Certain mobile apps** (e.g., Install and configure the Warehousing app overview and Project time entry mobile workspace) will be unavailable due to the Google Play Store not being available in China; however, alternatives are being considered.

- The **mobile platform** will not be available because certain App store dependencies are unavailable in China.
The following Microsoft Dynamics Lifecycle Services (LCS) features will have a different experience or will be unavailable due to the dependencies that are not available:

- **APQC Business process modeler (BPM) Library** will be unavailable. However, base Business process modeler (BPM) functionality will be available for custom models in April 2019. Search functionality in the BPM will be unavailable in China.

- **Electronic reporting (ER) overview assets** will not be available automatically, but can be manually uploaded from the LCS global asset library.

- **Code upgrade** will be unavailable for upgrades from Dynamics AX 2012.

- **Service and Support requests** will be available through LCS but 21Vianet is the service operator. For more information, see Support for Dynamics 365 Finance and Operations apps operated by 21Vianet in China.

- **Extensibility requests** will be unavailable.

- **Hotfix requests** will be unavailable.

- **Dynamics 365 Translation Service overview** will not be available.

- **Embedded Power Apps** and connectivity to Microsoft Power Apps and Microsoft Power Automate will be unavailable.

- **Integrate data into Microsoft Dataverse** will be unavailable.

---

**NOTE**
The Lifecycle Services URL for implementations operated by 21Vianet in China is lcs.dynamics.cn.

---

The following features will not be available due to certain current Azure Active Directory limitations in China:

- The **System administration > Setup > B2B Invitation configuration** page will not be available due to business-to-business (B2B) being unavailable in Azure Active Directory in China. For more information, see What is guest user access in Azure Active Directory B2B.

- **Conditional access** is an Azure Active Directory feature that is available for the Azure Active Directory Premium 2 SKU. This is unavailable in China.

- The Microsoft Dynamics 365 Payment Connector for PayPal is not available in China.

---

**Additional resources**

- [Dynamics 365 support site for 21Vianet (Chinese)](https://support.dynamics.com/cn)
- [Support for Dynamics 365 Finance and Operations apps operated by 21Vianet in China](https://support.dynamics.com/cn)
- [Model-driven apps in Dynamics 365 - operated by 21Vianet in China](https://support.dynamics.com/cn)
- [Dynamics 365 Privacy statement (Dynamics 365 隐私声明)](https://support.dynamics.com/cn)
- [Dynamics 365 Service Level agreement (世纪互联在线服务的 服务级别协议)](https://support.dynamics.com/cn)
- [Dynamics 365 Legal information (Dynamics 365 法律信息)](https://support.dynamics.com/cn)
- [Service terms for Dynamics 365 Lifecycle Services](https://support.dynamics.com/cn)
- [OSPT of Dynamics 365 (世纪互联 服务的 服务级别协议)](https://support.dynamics.com/cn)
- [Azure China 21Vianet](https://azure.cn)
- [Business applications availability in China – operated by 21Vianet in China](https://support.dynamics.com/cn)
Microsoft Dynamics 365 Finance, Dynamics 365 Commerce, and Dynamics 365 Supply Chain Management are generally available in the France geography. These deployment options serve customers in regulated industry and commercial organizations that do business with entities in France that require local data residency.

The deployment of the Dynamics 365 services in France is built on the foundational principles of security, privacy, compliance, transparency, and reliability. This deployment gives customers in France a complete cloud infrastructure and platform that provides familiar productivity and business application tools. Therefore, customer data stays resident in France.

**Provisioning**

The first prerequisite for deployment to France is to use the localized version of Microsoft Dynamics Lifecycle Services (LCS) that is referred to as **Go Local LCS, FR.LCS.Dynamics.com**. If you're interested in local deployment to a specific region, contact Microsoft to learn more about the onboarding process and the process for adding customers and partners.

**Features that aren’t available**

Microsoft strives to maintain functional parity between our commercially available service and Finance, Commerce, and Supply Chain Management in France. However, there are notable exceptions that are affected by dependent service or partner solution availability, market priorities, or compliance regulations. For more information about these exceptions, or for questions about services in France, contact **Microsoft Dynamics 365 Support**.

- Dynamics Regulatory Alert Submission service isn't available in France because it operates as a single global service. However, you can access the service from **Global LCS**.
- No corporate libraries are available in the American Productivity & Quality Center (APQC) Business process modeler (BPM) library. For each Go Local region, LCS published only the Getting started library and the last published APQC library. You can copy libraries to your project library to edit and publish them as the corporate library.
- The Dynamics 365 Translation Service overview isn't available in Go Local LCS. However, you can access the service from Global LCS.
- Electronic reporting (ER) assets aren't visible in the Shared asset library. You can manually upload the assets from the LCS Global asset library. To work around this issue, you can run a script that Microsoft Engineering provides. Alternatively, you can access Microsoft Support ER assets by connecting to the Global repository. For more information, see **Regulatory Configuration Service (RCS) - Lifecycle Services (LCS) storage deprecation**.
- The embedded Power BI dashboard isn’t available.
- Environment monitoring of LCS doesn’t expose the same dashboard interface as Global LCS. However, the core key performance indicators (KPIs) are available.
- The production environment update cadence for automatic updates is missing in the user interface (UI).
- For an Azure build pipeline that uses hosted agents: The nuggets package from the LCS library isn’t available in the Shared asset library.
- Environments are fully managed by Microsoft and don't support all customer self-service. Customers might have to create service requests for the Engineering team in LCS.
- Cloud scale unit for commerce, warehouse, and manufacturing isn’t available.
- Planning optimization service isn't available.
- Local business data deployment isn't available.

Additional resources

For information and links to resources that can help you set up legal entities that have a primary address in France, see France.

For information about product availability per country and workload, see Dynamics 365 and Power Platform availability.
This topic provides links to topics about development.

Overview

The Finance and Operations apps enable the entire enterprise resource planning (ERP) application suite as a cloud-based solution, for both public and private clouds, as well as on-premises. The apps leverage the speed, simplicity, and cost-effectiveness of working in the cloud, while building on the latest technology from Microsoft. The development experience includes:

- Development tools that are decoupled from any running environment. You develop against local, XML-based files, not the online database.
- Microsoft Visual Studio is the development environment. The Visual Studio environment is customized to provide you with a smooth and familiar experience.
- The X++ compiler generates Common Intermediate Language (CIL) for all features. CIL is the same intermediate language used by other .NET-based (managed) languages, such as the C# programming language.
- You can leverage the browser-based client and the design patterns for forms to provide an improved end-user experience.
- The Application Lifecycle Model (ALM) supports build automation, test automation, and deployment of models to the cloud.

Architecture

- Application stack and server architecture

Getting started

- Get evaluation copies
- Sign up for preview subscriptions
- Deploy and access development environments
- Development system requirements
- Removed or deprecated features
- Deprecated APIs
- Rename a local development (VHD) environment
- Introduction to Azure DevOps (Video)

Fleet Management

- Fleet Management sample application
- End-to-end scenario for the Fleet Management sample application

Development tools

Tutorials for development tools

- Development tools tutorial
- Create models and data model elements overview
- Build and debug projects
- Version control, metadata search, and navigation

**Tools, models, and VMs**
- Development tools in Visual Studio
- Application Explorer
- Finance and Operations project type in Visual Studio
- Element designers
- Commands for determining how elements are used
- Models and packages
- Build operations
- Code editor features
- Tools add-ins for Visual Studio
- Export and import models
- Metadata search in Visual Studio
- Create new users on development machines
- Update the Visual Studio development tools
- Development and build VMs that don't allow admin access FAQ

**Build automation using Azure**
- Build automation using Microsoft-hosted agents and Azure Pipelines
- Add license files to a deployable package in Azure Pipelines
- Create deployable packages in Azure Pipelines
- X++ model-versioning in Azure Pipelines
- Download assets by using Azure Pipelines
- Upload assets by using Azure Pipelines
- Deploy assets by using Azure Pipelines
- Create a Lifecycle Services (LCS) connection in Azure Pipelines
- Update the hosted Azure Pipeline for new NuGet packages
- Update a legacy pipeline in Azure Pipelines

**X++ programming language**

**Reference**
- X++ language reference

**Overviews**
- Write business logic by using C# and X++ source code
- Visual Studio requirements for X++

**Language support**
- EventHandlerResult classes in request or response scenarios
- Debug X++ code by using the debugger in Visual Studio
- Language Integrated Query (LINQ) provider for C#
- Write best practice rules
- SysSetupConfigAttribute attribute
Customize with extensions and overlayering
- Extensibility home page
- Customize App Suite reports by using extensions

Code migration
- How to resolve conflicts in Visual Studio (video)
- Code migration and upgrade home page

Move packages between environments
- Create deployable packages of models

Performance
- Take traces by using Trace parser
- Diagnose issues and analyze performance by using Trace parser
- Performance timer

User interface concepts
The client is an HTML web client that runs in all major browsers. For information about developing and customizing the user interface, see the User interface development home page.

Analytics
- Analytics, aggregate measurements, and KPI modeling

Reporting services
- Electronic reporting (ER) overview

Data entities and OData
- Data entities overview
- Open Data Protocol (OData)

Testing support in Visual Studio
- Testing and validations
- Test projects in Visual Studio
- Deploy and use a continuous build and test automation environment
- Task recorder resources

Office integration
- Office integration overview

Intelligence
- Business intelligence (BI) and reporting home page
Mobile platform
- Mobile platform resources

Global finance management
- Development for Dynamics 365 Finance home page

Development for independent software vendors
- Independent software vendor (ISV) development home page

Supply Chain Management
- Gantt control development guide
- Create a new transportation management engine

Additional resources
Insider tips on development
The application stack is divided into platform models and application-specific models. The platform models are Application Platform, Application Foundation, and Test Essentials. There are many application-specific models. Some examples are Application Suite, Ledger, Retail, and Case Management.

Overview

The application stack and server architecture align with three key pillars:

- **New client**
- **Cloud readiness**
- **New development stack**

The application stack is divided into several models: Application Platform, Application Foundation, Test Essentials, and the application suites. The separation enables new application development on the base foundation models, just as the Fleet Management sample application has been developed. Note the following important points about the changes in the server architecture:

- The services endpoint on the server is now responsible for returning all form and control metadata and data to the browser-based client. There is no longer any remote procedure call (RPC)-based communication with the server. The form objects still run on the server, and rendering has been optimized for browsers and other clients through server and client-side (browser) investments.

- The server, including the application code base, is deployed to an Internet Information Services (IIS) web application. In the cloud, it’s deployed to Microsoft Azure infrastructure as a service (IaaS) virtual machines (VMs).

- It is hosted on Azure and is available for access through the Internet. A user can use a combination of clients and credentials to access it. The recommended primary identity provider is OrgID, and the store for the identity is Azure Active Directory (Azure AD). The security subsystem uses the same AuthZ semantics for users and roles.

- Two types of clients must be considered for access in the cloud: active clients and passive clients.
  - Active clients can programatically initiate actions based on responses from the server. An active client doesn't rely on HTTP redirects for authentication. A smart/rich client is an example of an active client.
  - Passive clients can't programatically initiate actions based on responses from the server. A passive client relies on HTTP redirects for authentication. A web browser is an example of a passive client.

Currently, Access Control Service (ACS) doesn't support a mechanism for non-interactive authentication. Therefore, even when active clients try to authenticate by using ACS, they must use passive client authentication, in which a browser dialog box prompts the user to enter their credentials.

- A completely revamped metadata subsystem incorporates the new compiler and Microsoft Visual Studio-based development model. The model store is represented as a set of folders and XML artifacts that are organized by model. The model elements, such as tables, forms, and classes, are represented by an XML file that contains both metadata and source code.

The left side of the following diagram shows how the application stack has been split into distinct models. The right side shows how the key components are stacked in the server.
The Finance and Operations applications use an entry point security model. A form allows read-only access if the menu item used for navigation to that form has only Read Permissions. However, navigation to that same form through another menu item that provides Create Permissions, Delete Permissions, or Update Permissions allows write operations on the form. This behavior simplifies the development experience, because developers can specify the behavior for a form through a given entry point.

Cloud architecture

The cloud architecture includes services that automate software deployment and provisioning, operational monitoring and reporting, and seamless application lifecycle management. The cloud architecture consists of three main conceptual areas:

- **Lifecycle Services (LCS)** – LCS is a multi-tenant shared service that enables a wide range of lifecycle-related capabilities. Capabilities that are specific to this release include software development, customer provisioning, service level agreement (SLA) monitoring, and reporting capabilities.
- **Finance and Operations** – The VM instances are deployed through LCS to your Azure subscription. Various topologies are available: demo, development/test, and high-availability production topologies.
- **Shared Microsoft services** – A Finance and Operations application uses several Microsoft services to enable a “One Microsoft” solution where customers can manage a single sign-in, subscription management, and billing relationship with Microsoft across Finance and Operations applications, Microsoft 365, and other online services.

Many features of the Azure platform are used, such as Microsoft Azure Storage, networking, monitoring, and SQL Azure, to name a few. Shared services put into operation and orchestrate the application lifecycle of the environments for participants. Together, Azure functionality and LCS will offer a robust cloud service.

Development environment

The architecture of the development environment resembles the architecture of the cloud instance. It also includes the software development kit (SDK), which consists of the Visual Studio development tools and other components. Source control through Team Foundation Server or Visual Studio Online enables multiple-developer scenarios, where each developer uses a separate development environment. Deployment packages can be compiled and generated on a development environment and deployed to cloud instances by using LCS. The following diagram shows how the key components interact in a development environment.
A public preview is available. You can sign up and deploy a cloud instance of the latest build. This public preview available through Microsoft Dynamics Lifecycle Services (LCS). These links provide more information about how to download and use the public preview:

- Sign up for preview subscriptions
- Service update availability
- Partner Trial
- How can I setup a solution trial instance in Azure with my customization and demo data?
- How do I login to the new AX as a demo user persona?
This topic explains how to subscribe to the preview/partner offer and deploy an environment. The subscription that you create gives you a Microsoft Online Services test tenant and a Microsoft Dynamics Lifecycle Services (LCS) project where you can deploy an environment. This topic will also help you set up additional users in your Microsoft Online Services tenant and gain experience with service administration capabilities. Here are the skills that you will learn:

- Subscribe, and create a new Microsoft Online test tenant.
- Navigate to LCS projects.
- Use various features of LCS.
- Add users to Microsoft Azure Active Directory (Azure AD) and the client.
- View resources in your subscription email.

Key terms

- **Microsoft Online Services tenant** – A tenant is the group of all subscriptions and users for your organization. The tenant is created at the same time as your first subscription in Microsoft Online Services.

- **Subscription** – A subscription gives you an online cloud environment and experience. It also lets you see how customizations that you develop can be deployed to the cloud.

- **Microsoft Azure Active Directory** – The cloud environment includes Azure AD. Azure AD helps you manage users, groups, security roles, and licenses for online applications, much as you manage them for on-premise environments.

- **Users** – Users of the services that your organization has subscribed to are managed in Azure AD. Any users in your tenant can be added and assigned to security roles.

- **Developers and administrators** – Developers and administrators are users who also have access to LCS that lets them manage projects and environments. These users are also end users.

- **Organizational account** – Users receive Azure AD credentials. These credentials are separate from other desktop or corporate credentials. The Azure AD credentials are used to sign in to Microsoft 365 and other Microsoft cloud services. Users sign in by using their organizational account.

**IMPORTANT**

For this release, we ask that you not use any existing credentials that are associated with other online services, such as Microsoft 365 or Microsoft Dynamics CRM Online.

- **Microsoft account** – Microsoft accounts were formerly known as Passport accounts or Windows Live ID accounts. Currently, Microsoft accounts can’t be used with Finance and Operations applications, Microsoft Dynamics 365 Commerce, or other Microsoft Online Services. However, Microsoft accounts are still required for Microsoft Connect and other Microsoft Business Solutions sites, such as Information Source and Microsoft Dynamics Community. You will continue to use your Microsoft account to access these services.

- **Microsoft 365 admin center** – Microsoft 365 admin center is the subscription management portal
that Microsoft 365 provides for administrators. Microsoft 365 admin center is used to provide management functions for users and subscriptions.

- **Environments** – You can deploy as many single instances of a virtual machine (VM) as you require. We call these instances *environments*.

**Prerequisites**

1. You've received an email that invites you to participate in the preview.
2. If your company has an organizational account with Microsoft Online Services, and you're signed in, you must sign out before you continue. Alternatively, you can use *InPrivate Browsing* mode.
3. If you aren't sure whether you're signed in, delete your browser cookies, and then close your browser before you continue.

**Subscribe**

1. Finance and Operations applications and Retail are available only to existing Microsoft Dynamics 365 channel partners and customers who are currently enrolled in the Business Ready Enhancement Plan (BREP) service plan. To subscribe, visit [PartnerSource Business Center](https://partnersource.microsoft.com).
2. On the *Account setup* page, in the *Country or region* field, select the country or region.
3. Follow the wizard and prompts to complete the sign-up, until you reach the last step.

**You’re ready to go...**

**Start a new project in LCS**

To use LCS to manage your environments, you must create a new project.

1. Go to [https://lcs.dynamics.com/Logon/Index](https://lcs.dynamics.com/Logon/Index).
2. Select *Sign in*.
3. Sign in by using the account that you used to subscribe.
4. Select the plus sign (+) to create a new project.
Add users to LCS

You’re already set up as a user of your LCS project. If you’ve also added other Microsoft 365 users, you must add them to this project. Other administrators and developers will then be able to deploy their own environments. These LCS users are team members who will actively work on the implementation. Don’t confuse them with end users. Start on the project page in LCS.

1. Scroll to the right, and then, in the More tools section, select the Project users tile.
2. In the upper left, select the plus sign (+) to add a new user.
3. In the Email field, enter the email address of the user that you’re adding. This email address should be the Microsoft 365 organization email address that you created earlier.
4. In the Project role field, select Project Owner.
5. Select Invite.
6. Repeat steps 2 through 5 for all users in your organization.
Deploy environments

Environments should be deployed to an existing Azure subscription.

NOTE

Each developer of an environment must deploy their own system to Azure. However, only the first project user must set up the Azure subscription for deployment.

You can create environments in two ways:

- Deploy to Microsoft cloud services (Azure).
- Download a local virtual hard disk (VHD).

Start on the project page in LCS.

1. In the Environments section, select the plus sign (+). The Microsoft Azure setup dialog box appears.
2. Enter your Azure subscription ID. You can find your Azure subscription ID in Azure Portal (https://ms.portal.azure.com/), under Settings in the lower left.
3. Select Next.
4. Download the Azure Management Certificate to a local folder on your computer, and then upload it to Azure Management Portal by going to Settings > Management Certificates. This certificate will enable LCS to communicate with Azure on your behalf.
5. Return to LCS, and select Next.
6. Select the Azure region to deploy in. The West US region will have the fastest deployments, but it's important that you select a data center that is close to where you plan to use this system.
7. Select Connect.
8. In the list of available topologies, select the topology to deploy. You can select either the Download link to download the VHD or Next to deploy on Azure. Azure is the preferred path.
9. Enter the name of the environment.
10. Read the pricing and licensing terms, and then select the check box to indicate that you understand them.
11. Select Next.
12. Confirm the details, and then select Deploy.

NOTE

Developers and administrators who will use their own environments must sign in and repeat these steps.

After you deploy your environment, it's available in the Environments section.
13. Select the environment to view details about the deployment status. The first deployment will require a few hours, but each subsequent deployment will be much faster.

14. When the deployment status changes to **Deployed**, select **Login** to connect to the client, or select the name of the VM to connect to the development machine by using Remote Desktop. After the deployment is completed, you can find the base URL and the information that you require to connect to the environment via Remote Desktop.

**Use the features of LCS**

LCS is the starting point for performing online administrative activities. Here are some of these activities:

- Deploy VMs on Azure.
- Access materials.
- Access downloads of tools and resources.

**Explore the LCS project**

1. Review the methodology, and complete the tasks and phases as you progress through the life cycle. The phases and task information lets you view tools and resources that are available throughout your enterprise resource planning (ERP) experience.

2. Scroll to the right, and review the tiles.
The available tiles include various tools and services in LCS. They also include the following additional tiles:

- **My subscription** – The Microsoft 365 subscription management portal is where you can view and work with your online subscriptions. By selecting **User and Groups** in the left navigation section of the page, you can also manage your online users.

  
  **NOTE**
  
  To access the page, you must be a member of the **Global Administrator** role for your organization's Microsoft Online Services tenant.

- **Feedback and bugs** – This tile opens the **General Feedback** page in Microsoft Connect. Use this page to record bugs, and to design change requests, feature requests, and suggestions.

- **Microsoft 365 users** – This tile opens the **Users and groups** page in Microsoft 365 admin center. You can add, update, and remove users, reset passwords, and assign licenses for other services.

  
  **NOTE**
  
  To access the page, you must be a member of the **Global Administrator** role for your organization's Microsoft Online Services tenant. The installing user is always a global administrator, but other users must be added to this role.
This topic describes how to access development instances, configure local development virtual machines (VMs), and find important configurations settings for developers and administrators.

**NOTE**
- Microsoft Support may provide limited troubleshooting on Tier 1 development environments.
- In certain circumstances, a fresh deploy of a Tier 1 environment may be requested by Microsoft Support to resolve an issue.
- Development environments should not contain business critical data and are considered disposable.

### Definitions

<table>
<thead>
<tr>
<th>TERM</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>End user</td>
<td>A user who accesses an instance through the web client. The end user must have Microsoft Azure Active Directory (Azure AD) credentials to access an instance and must be provisioned/added as a user of that instance.</td>
</tr>
<tr>
<td>Developer</td>
<td>A user who will develop code through the Microsoft Visual Studio environment. A developer requires Remote Desktop access to development environment (VM). The developer account must be an administrator on the VM.</td>
</tr>
</tbody>
</table>

### Deploying cloud development environments

To deploy a cloud development environment in your LCS project:

1. Create a connection between an LCS project and your Azure subscription. You will need your Azure subscription ID and authorize the use of the subscription.

2. Select + under **Environments** to deploy.
3. Select an application and platform version.

4. Select an environment topology. For more information, see Sign up for preview subscriptions.

5. If you chose a cloud-hosted environment, select which Azure connector you want to use. Then select Deploy.

When a cloud environment is provisioned through LCS:

- The user who requests the cloud environment is provisioned as the administrator in that environment.
- User accounts are provisioned on the development VM to allow access to the environment using Remote Desktop, these credentials are accessible on the environment page in LCS.
Accessing an instance through a URL

The system can be accessed by end users. The administrator can add users to this system by using the Users page in the instance. Note that these additional users don't have to be users in LCS. You obtain the base URL for the cloud environment from your LCS project site.

1. Go to your LCS project navigation menu, and select Cloud-hosted environments.
2. In the environment list section, select the deployed environment.
3. When the environment page opens, you can access the application by clicking Login > Log on to Finance and Operations in the upper-right corner.
4. Use valid end user credentials to sign in to the application. If the current LCS user is the user who originally deployed the environment, that user is probably a valid end user and the administrator of the application.
5. In your browser, make a note of the base URL after you sign in. For example, the base URL might be https://dynamicsAx7aosContoso.cloud.dynamics.com.

Accessing the cloud instance through Remote Desktop

Cloud environments can be accessed both as an end user and as a developer. The developer gets access to the system through Remote Desktop credentials. The Remote Desktop credentials are obtained from the environment page on the LCS project site (see the illustration earlier in this topic).

For environments deployed before Platform update 12:

1. Click the VM name.
2. Use the local administrator user name and password that are shown to connect to the cloud VM through Remote Desktop. You can reveal the password by selecting the show password icon.

For any environments deployed on or after Platform update 12, there are distinct accounts, a developer account and an admin account.

After you sign in to the environment through Remote Desktop, if you want to access the local application from the browser, use the same base URL that you use to access the application from a remote computer. The previous section explains how to obtain this base URL from LCS.

VM that is running locally

A virtual hard disk (VHD) is made available for download from LCS, so that you can set it up on a local machine. This system is intended to be accessed by a developer and is a pre-configured one-box development environment of Finance and Operations apps. The VHD is available in the Shared Asset library of LCS under the asset type Downloadable VHD.

1. Go to the LCS main page and select Shared asset library or go to Shared Asset Library.
2. Select the asset type Downloadable VHD.
3. Find the VHD you are looking for based on the desired Finance and Operations version. The VHD is divided into multiple file parts that you need to download. For example, the asset files that start with "VHD - 10.0.5"
are the different files you need in order to install version 10.0.5.

4. Download all files (parts) associated with the desired VHD to a local folder.

5. After the download is complete, run the executable file that you downloaded, accept the software license agreement, and choose a file path to extract the VHD to.

6. This creates a local VHD file that you can use to run a local virtual machine.

**Commerce configuration**

Follow the steps in this section if you are also configuring for Commerce.

To use the downloadable VHD for POS customizations, you must also follow this step.

- On the host computer, enable Nested VM support. For more information, see Run Hyper-V in a Virtual Machine with Nested Virtualization.

**Running the virtual machine locally**

Follow these steps to run the VM from Hyper-V Manager.

1. To start the VM, select **Start**.

2. To open the VM in a window, select **Connect**.

3. Select the **Ctrl+Alt+Delete** button on the toolbar. The VM receives most keyboard commands, but Ctrl+Alt+Delete isn’t one of them. Therefore, you must use the button or a menu command.

4. Sign in to the VM by using the following credentials:
   - **User name:** Administrator
   - **Password:** pass@word1

   **TIP**

   You can resize the VM window by changing the screen resolution. Right-click the desktop on the VM, and then click **Screen resolution.** Select a resolution that works well for your display.

5. Provision the administrator user. For more information, see the next section.

6. Start the Batch Manager Service. This step is required if you’re running batch jobs or workflows.
   - a. Open a **Command Prompt** window as an administrator.
   - b. Type `net start DynamicsAxBatch`, and then press Enter.
      You can also start the service from the **Services** window.

7. **Apply updates** as needed.

**Commerce configuration**

For POS customizations, you must also follow these steps on the guest VM.

1. Download and install Microsoft Emulator for Windows 10 Mobile Anniversary Update.

2. Start the Hyper-V host service. For more information, see Hyper-V: The Hyper-V Virtual Machine Management service must be running. If errors occur during startup, you can also try to uninstall and reinstall the Hyper-V role on the guest VM.

**Provisioning the administrator user**

For developer access, you must be an administrator on the instance. For environments that are provisioned through LCS, we encourage you to deploy with the correct user. For more information, see Frequently asked questions. To provision your own credentials as an administrator on a local VM, run the Admin user provisioning tool. On the local VM, a link is provided on the desktop.
1. Run the admin user provisioning tool as an administrator (right-click the icon, and then click Run as administrator).

2. Enter your email address, and then select Submit.

**NOTE**
The Admin user provisioning tool isn't supported in environments that are provisioned through LCS. It should be used only on local VMs.

---

**Commerce configuration**

Follow the steps in this section if you are also configuring for Commerce.

*For Dynamics 365 for Operations, Version 1611*

1. Run the RetailTenantUpdateTool.
   - The icon for this tool is available on the desktop.
   - This tool is also available at the following location:
     `C:\windows\System32\WindowsPowerShell\v1.0\PowerShell.exe -File C:\RetailSDK\Tools\RetailTenantUpdateTool.ps1`

2. Double-click the icon to start this tool. You will be prompted for your Azure AD credentials. You must use the same credentials that you used in the admin user provisioning tool earlier.

   ```powershell
   $msocred = Get-Credential
   Connect-MsolService -Credential $msocred
   $company = Get-MsolCompanyInformation
   Write-Host "TenantID: $($company.ObjectId)"
   $msocred.UserName
   $users = Get-MsolUser -SearchString "($($msocred.username)"
   foreach($u in $users)
   {
       if($u.SignInName -eq $msocred.UserName)
       {
           Write-Host "SignInName:($u.SignInName) UserId: $u.ObjectId"
       }
   }
   ```

*For Dynamics 365 for Operations 7.0*

1. Install Microsoft Online Services Sign-In Assistant for IT Professionals RTW.

2. Install Azure Active Directory Module for Windows PowerShell (64-bit version).

3. Query Azure AD for your tenant and user ID. Open a Windows PowerShell Integrated Scripting Environment (ISE) window with administrative privileges, and run the following command. You will be prompted for Azure AD credentials. Use the same user account that you used in the admin user provisioning tool earlier.

   ```powershell
   $msocred = Get-Credential
   Connect-MsolService -Credential $msocred
   $company = Get-MsolCompanyInformation
   Write-Host "TenantID: $($company.ObjectId)"
   $msocred.UserName
   $users = Get-MsolUser -SearchString "($($msocred.username)"
   foreach($u in $users)
   {
       if($u.SignInName -eq $msocred.UserName)
       {
           Write-Host "SignInName:($u.SignInName) UserId: $u.ObjectId"
       }
   }
   ```
4. Update the following SQL script, and run it in on AXDB for that environment. Supply values for the following parameters from the preceding Windows PowerShell script output:

- **TenantID** – For example, c83429a6-782b-4275-85cf-60eb81250ee
- **UserId** – For example, a036b5d8-bc8c-4abe-8eec-17516ea913ec

```sql
DECLARE @TenantId NVARCHAR(1024)         DECLARE @UserId NVARCHAR(1024)
SET @TenantId = ''
SET @UserId = ''
IF(LEN(@TenantId) > 0 AND LEN(@UserId) > 0)
BEGIN
    UPDATE AxDBRAIN.dbo.SYSSERVICECONFIGURATIONSETTING SET [VALUE] = @TenantId WHERE [NAME] = 'TENANTID'
    UPDATE RetailHoustonStore.ax.SYSSERVICECONFIGURATIONSETTING SET [VALUE] = @TenantId WHERE [NAME] = 'TENANTID'
    UPDATE AxDBRAIN.dbo.RETAILSTAFFTABLE SET EXTERNALIDENTITYID = @TenantId, EXTERNALIDENTITYSUBID = @UserId WHERE STAFFID = '000160'
END
ELSE
BEGIN
    RAISERROR (15600, -1, -1, 'TenantId and UserId must be set before running this script')
END
```

5. Reset Internet Information Services (IIS) by running `IISRESET` in an elevated **Command Prompt** window.

6. Update the Real-time service profile to use the new admin user.

   a. Go to **Retail and Commerce > Headquarters setup > Commerce scheduler > Real-time service profiles**.
   b. Edit the JBB record so that it uses the user that you used earlier (for example, `administrator@contosoax7.onmicrosoft.com`).
   c. Run CDX Job 1070 (Staff) for the default channel database.
   d. Verify that the job succeeded by viewing the **Download Sessions** page on the client.

**Base URL of the local application**

After the user is provisioned as an administrator, that user can access the instance on the computer by navigating to the following base URL: [https://usnconeboxax1aos.cloud.onebox.dynamics.com](https://usnconeboxax1aos.cloud.onebox.dynamics.com). If you're using version control and plan to connect multiple development VMs to the same Azure DevOps project, rename your local VM. For instructions, see **Rename a local development (VHD) environment**.
The URL of the POS app is https://usnconeboxax1pos.cloud.onebox.dynamics.com/. After you complete the configuration steps, this VM is provisioned with your Azure AD tenant. Your Azure AD admin account is mapped to a cashier worker account in demo data. You can use this cashier account to easily activate a POS device in this environment.

- Cashier user ID: 000160
- Cashier password: 123
- Cashier LE: USRT
- Cashier store: Houston

Location of packages, source code, and other AOS configurations

On a VM, you can find most of the application configuration by opening the web.config file of AOSWebApplication.

1. Start IIS.
2. Go to Sites > AOSWebApplication.
3. Right-click, and then click Explore to open File Explorer.
4. Open the web.config file in Notepad or another text editor. The following keys are of interest to many developers and administrators:
   - **Aos.MetadataDirectory** – This key points to the location of the packages folder that contains platform and application binaries, and also source code. (Source code is available only in development environments.) Typical values are: c:\packages, c:\AosServicePackagesLocalDirectory, and J:\AosServicePackagesLocalDirectory.
   - **DataAccess.Database** – This key holds the name of the database.
   - **Aos.AppRoot** – This key points to the root folder of the Application Object Server (AOS) web application.

Commerce configuration

The software development kit (SDK) is available at C:\RetailSDK. For more information about how to use and customize applications, see the following topics:

- Retail software development kit (SDK) architecture
- Point of sale (POS) device activation

Redeploying or restarting the runtime on the VM

To restart the local runtime and redeploy all the packages, follow these steps.

1. Open File Explorer, and go to C:\CustomerServiceUnit.
2. Right-click AOSDeploy.cmd, and then click Run as administrator.

This process might take a while. The process is completed when the cmd.exe window closes. If you just want to restart AOS (without redeploying the runtime), run **iisreset** from an administrator Command Prompt window, or restart AOSWebApplication from IIS.

Frequently asked questions

Environment is in a failed state and the error message is "Updated AAD Tenant is missing reply URL"
configuration
This message indicates that a Tier 1/customer-managed environment is configured with an Azure AD tenant that is different than the tenant used at the time of deployment. (Perhaps an update was done using the Admin user provisioning tool.) The updated tenant currently being used is missing the reply URL configuration required for successful login into the environment. The missing configuration causes the error. You should delete the environment and redeploy with a user from the tenant that the environment will be used with.

As a partner/ISV, how can I facilitate cloud-hosted deployments for customers that I work with?
A Tier 1/customer-managed environment should be deployed under the customer's Azure AD tenant, to ensure that all the configuration and integrations are correctly provisioned for any given environment. The tenant and environment association is determined based on the user who deployed the environment.

To facilitate cloud-hosted deployments, we recommend that partners follow this step to create customer-specific, cloud-hosted environments. In this way, they will ensure that the deployment is registered under the correct tenant.

- Deploy the environment via a user from the tenant that the environment will be used with. The Admin user provisioning tool should not be used to change the tenant for a Tier 1/customer-managed/cloud-hosted environment.

NOTE
The Azure AD tenant that is associated with the Azure subscription doesn't play any role in environment configuration. The Azure subscription and the corresponding connector configuration are used only to deploy Azure resources.

I have run the Admin user provisioning tool on my development environment, and now I receive the following sign-in error: "Error: AADSTS50011: The reply URL specified in the request does not match the reply URLs configured for the application."
As was stated earlier, it's very important that Finance and Operations environments be deployed under the correct Azure AD tenant. For Tier 1/customer-managed environments that are deployed via LCS, changes to the Azure AD tenant settings aren't supported after deployment.

How can I fix my existing environment when my environment is in a failed state or I am getting sign-in errors?
If you have environments where the Admin user provisioning tool was previously used to update the tenant settings, we recommend that you delete those environments and then redeploy them under the correct Azure AD tenant.

If an existing environment can't be deleted and redeployed, its URL must be added to the configured Azure AD tenant. The following commands can be run by the tenant admin.

1. Retrieve the following values from the web.config file.

```
$AADTenant = <Value of Aad.TenantDomainGUID from web.config>
$EnvironmentUrl = <Value of Infrastructure.HostUrl from web.config>

# For example, if value is spn:fd663e81-110e-4c18-8995-ddf534bcf5e1 then take only fd663e81-110e-4c18-8995-ddf534bcf5e1
$AADRealm = <Value of Aad.Realm from web.config without spn: prefix.>
```

2. Run the following commands via the tenant admin account for the Azure AD tenant in the web.config file.
# Using tenant admin account under this tenant login to via AzureAD PowerShell cmdlet.
Connect-AzureAD

# Get Service Principal details
$SP = Get-AzureADServicePrincipal -Filter "AppId eq 'AADRealm'"

# Add Reply URLs
$SP.ReplyUrls.Add("$EnvironmentUrl")
$SP.ReplyUrls.Add("$EnvironmentUrl/oauth")

# Set/Update Reply URL
Set-AzureADServicePrincipal -ObjectId $SP.ObjectId -ReplyUrls $SP.ReplyUrls
This article describes recommended configurations of your one-box developer environment.

**Setup**
1. Start Visual Studio, and on the toolbar, click **Dynamics 365 > Options**.
2. Expand the **Microsoft Dynamics** node, and then click **Projects**.
3. Verify that the **Organize projects by element type** check box is selected, and click **OK**.
4. To view the line numbers in your code editor, select **Tools > Options > Text Editor > All Languages**.
5. Select the **Line numbers** check box.

**Debugging**
For better performance of the X++ debugger, you might want to turn off IntelliTrace. IntelliTrace collects the complete execution history of an application. It is not supported for X++ debugging and causes performance issues in the IDE when debugging large packages like Application Suite. To turn off Intellitrace, click **Options > IntelliTrace > Enable IntelliTrace**, clear the check box, and then click **OK**. Note that Intellitrace is only available in the Enterprise version of Visual Studio.
This topic provides answers to frequently asked questions (FAQ) about virtual machines (VMs) that don't allow administrator access.

How can I install a deployable package?

Whenever possible, use Microsoft Dynamics Lifecycle Services (LCS) to install a deployable package. You can install a deployable package by using the `devinstall` option. Remember that this option requires manual database synchronization.

For more information about how to install a deployable package, see Install deployable packages from the command line.

Is the Finance and Operations website accessible when Visual Studio isn't running?

Yes, you can access the Finance and Operations website when Microsoft Visual Studio isn't running. Microsoft Internet Information Services (IIS) Express is an .exe file that runs as the user. However, when you close Visual Studio, the XPPC agent starts regular IIS (not IIS Express) before it closes. This behavior helps to ensure that you can remotely access the Application Object Server (AOS) instance and the website, even when you sign out or the machine is restarted. We recognize that many people use these developer machines as test machines, and that they expect the AOS instance always to be running. However, IIS Express doesn't support this behavior.

What about the other services?

You can restart Microsoft Windows services such as Microsoft SQL Server, SQL Server Reporting Services (SSRS), SQL Server Integration Services (SSIS), SQL Server Analysis Services (SSAS), Batch, Financial reporting (formerly Management Reporter), and IIS. (For IIS, you must restart the World Wide Web Publishing Service because you can't use `iisreset.exe`.)

Can I clean up the service volume drive?

Yes, you have full access to the service volume drive. Therefore, you can clean up the monitoring data, and so on.

What are the alternatives to VMs that don't allow administrator access?

Both a Microsoft Azure environment on a private Azure subscription and a local virtual hard disk (VHD) allow administrator access. However, you must run Visual Studio as an administrator. This requirement applies because the administrator has access to these alternatives only through the `administrator` group, not explicitly.

Can I run Visual Studio as an administrator?

You are not required to run Visual Studio as an administrator. You cannot use the Remote Desktop Protocol (RDP) to connect as an administrator to VMs that are under a Microsoft-owned Azure subscription. These VMs include the Tier 1 VM that is included in the subscription and Tier 1 add-on VMs. However, if you're connecting
as an administrator to a VM that isn't under a Microsoft-owned subscription, you must still run Visual Studio as an administrator.

A "get latest" operation in Visual Studio failed because files are blocked by the AOS instance. How do I start and stop IIS?

You must use IIS Express. See the next question for more information.

What are the instructions for using IIS Express?

When IIS Express is started, an icon appears in the notification area (near the clock). When you right-click on the IIS Express icon, all the running sites are listed. You can stop IIS Express from that menu. Some actions in Visual Studio cause IIS Express to be started, but you can also explicitly start IIS Express from Visual Studio by selecting Restart IIS Express on the Dynamics 365 menu.

To ensure that debugging functions properly with IIS Express and Finance and Operations Visual Studio projects, we recommend the following Internet Options settings:

- Go to Control Panel > Internet Options > Security tab > Internet, and clear the Enable Protected Mode check box.
- Go to Control Panel > Internet Options > Security tab > Restricted sites, and clear the Enable Protected Mode check box.

Can I install additional development tools (such as Fiddler and Pepper)?

No, you can't install additional development tools.

Is there a way to run Windows PowerShell and command prompt commands as an administrator?

No, you can't run Windows PowerShell commands and commands at a prompt command as an administrator.

Is the Trace Parser supported?

Trace Parser currently requires the user to be an administrator. It is not supported on dev/test environments that are managed by Microsoft that do not allow administrator access.

Is the Admin user provisioning tool supported?

The Admin user provisioning tool currently requires the user to be an administrator. The Admin user provisioning tool is typically used to change the tenant of the environment, but that should not be necessary. You can update the sign in information in the database for the Admin user or any other user. You only need the SID and network alias (email address) from a user that can access the environment or another environment on the same tenant. In many cases, the SID and network alias can be found in the database that came with the environment originally. Run the following commands to get the good SID and network alias from the source environment and update them in the target environment, respectively.

```sql
-- get value from source env.
select ID, SID, NETWORKALIAS from USERINFO where ID = 'Admin'

-- update value in target env.
update USERINFO set SID = 'new_SID', NETWORKALIAS = 'new_NetworkAlias' where ID = 'Admin'
```
Can the system be put into maintenance mode?

You can put the system into maintenance mode to change the license configuration. However, the procedure that is described in Maintenance mode isn't supported. Self-service support for maintenance mode in all environments will be added to LCS in the future. Until this support is available in LCS, you can follow these steps to put a system into maintenance mode.

1. Establish an RDP connection to the developer machine.
2. On the developer machine, sign in to SQL Server by using the credentials for the axdbadmin user from LCS. Then switch to the AXDB database, and run the following command.

   ```
   update SQLSYSTEMVARIABLES SET VALUE = 1 where PARM = 'CONFIGURATIONMODE'
   ```

3. Restart the World Wide Web Publishing Service to reset IIS.

   After the service is restarted, the system will be in maintenance mode.

4. When you've completed your maintenance mode activities, repeat steps 2 and 3, but set the value to 0 in step 2.

Can I install a license deployable package?

**Development environments**

Use LCS to install a license deployable package on any cloud development environment.

**Build environments**

LCS does not allow AOT or license deployable packages to be installed on build environments. To work around this, remote into the VM and use the -devinstall option to install a license deployable package from the command line as described in the topic, Install deployable packages from the command line. This command line install works as of platform update 17. If you are running on a platform version that is older than Platform update 17, and you do not have admin access to your build environment, create a support request and ask Microsoft to install your license deployable package.

Is licensing Visual Studio by entering a product key supported?

Entering a product key directly in Visual Studio is not supported. Instead, use Visual Studio subscription licensing and sign in to Visual Studio with the email address (user account) associated with the license. You can link a Visual Studio license to a user account by assigning an MSDN license to the user account or by assigning a license to the user account by using https://www.visualstudio.com/subscriptions-administration.

Can I upgrade my database to a new application release?

As of the February 2018 release of Lifecycle Services (LCS), you can execute the data upgrade package from the LCS environment page of a development environment. Executing the data upgrade package from LCS does not require you to be an administrator on the VM.

The process described in Upgrade data in development or demo environments runs the data upgrade package from the command line. This requires you to be an administrator on the VM.

What do I need to know if I am developing for Commerce?

If you are developing for Dynamics 365 Commerce, configuration steps and other important information is described in Development in cloud-hosted development environments without admin access.
A local development (VHD) environment must be renamed for the following scenarios:

- **Accessing a single Microsoft Azure DevOps project across multiple machines:** Azure DevOps is required for version control. In development topologies, multiple virtual machines (VMs) can't access the same Azure DevOps project if they have the same machine name. Azure DevOps uses the machine name for identification. If you're developing on local VMs that were downloaded from Microsoft Dynamics Lifecycle Services (LCS), you might encounter issues.

- **Installing One Version service updates:** One Version service updates, such as 8.1.x, must be installed in VHD environments by using a runbook. To help guarantee that the runbook is completed successfully, the VHD environments must be renamed. Additional steps that are described in this topic must also be completed.

**Rename the machine**

Rename and restart the machine before you start development or connect to Azure DevOps. Make sure that the new name is unique among all the machines that are used with the Azure DevOps project.

**Update the server name in SQL Server**

Update the server name in Microsoft SQL Server 2016 by running the following commands.

```
sp_dropserver [old_name];
GO
sp_addserver [new_name], local;
GO
```

In these commands, be sure to replace `old_name` with the old name of the server and `new_name` with the new name. By default, the old name is `MININT-F36S5EH`, but you can run `select @@servername` to get the old name. Additionally, be sure to restart the SQL Server service after the commands have finished running.

**Update SQL Server Reporting Services**

Update the SQL Server Reporting Service (SSRS) database by using the Reporting Services Configuration Manager. Select **Database**, select **Change Database**, and use the new server name. Make sure that you use Reporting Services Configuration Manager for SQL Server 2016.

**Additional steps to install One Version service updates**

The following additional steps are required in order to install One Version service updates in a VHD environment.

- **Update the Azure Storage Emulator**

  Update the Azure Storage Emulator, and make sure that it's running. From the **Start** menu, open Microsoft Azure Storage Emulator - v4.0, and run the following commands.

  This command starts the emulator.
AzureStorageEmulator.exe start

This command verifies that the emulator is running.

AzureStorageEmulator.exe status

Try the `init` option with the `--server` switch or the `--forcecreate` switch. Be sure to replace `new_name` with the new name.

```
AzureStorageEmulator.exe init --server new_name
AzureStorageEmulator.exe init --forcecreate
```

If the `init` command fails, delete the storage emulator database by using SQL Server Management Studio. Then try the following command.

```
AzureStorageEmulator.exe init
```

When you run this command, you might receive the following error message: "Error: Cannot create database." However, the emulator will usually still start. You just need the emulator to start.

**Update financial reporting**

Update the server name for financial reporting by using a script that is included in the One Version service update. To get the command, you must download and expand the One Version service update.

Open a Microsoft Windows PowerShell command window as an admin, and run the following command. This command contains the default passwords that might have to be updated. Be sure to replace `new_name` with the new name.

```
\cd <update folder>\MROneBox\Scripts\Update
\ConfigureMRDatabase.ps1 -NewAosDatabaseName AxDB -NewAosDatabaseServerName new_name -NewMRDatabaseName ManagementReporter -NewAxAdminUserPassword AOSWebSite@123 -NewMRAdminUserName MRUser -NewMRAdminUserPassword MRWebSite@123 -NewMRRuntimeUserName MRUSer -NewMRRuntimeUserPassword MRWebSite@123 -NewAxMRRuntimeUserName MRUser -NewAxMRRuntimeUserPassword MRWebSite@123
```
This topic lists the system requirements for development.

Development environments can be hosted locally or in Microsoft Azure. The build process, X++ compilation, and generation of cross reference information, typically run satisfactorily on machines with 16 GB of memory and two CPU cores. The compiler uses available resources, so more RAM and more cores can speed up compilation, especially if there is contention for the resources from other concurrent processes. If you are running concurrent processes, then we recommend 24 GB of memory with four cores. At a minimum, two CPU cores are recommended because the developer environment contains many components that may be running concurrently. The components include the AOS web application, Visual Studio, Management Reporter, and SQL Server.
This tutorial will walk you through configuring Microsoft Azure DevOps to enable source control on your models. It will also help you learn about other productivity features in the development tools, including the ability to create and organize TODO task, search metadata and source code, navigate between related model elements, and create a project from a model.

**Configure your Azure DevOps organization and project**

In this section, you’ll create a new project in Azure DevOps. This project will host the source code of your model. You’ll use the Fleet Management model as an example. If you don’t have a Azure DevOps organization, you’ll create one.

**Sign up to Azure DevOps, create an account, and create a new project**

Navigate to [https://www.visualstudio.com/](https://www.visualstudio.com/) to sign up for Azure DevOps. Click Sign up. If you already have an account in Azure DevOps, go to the Create a Azure DevOps project section later in this topic.

1. Sign in with your Microsoft account.

   **NOTE**
   You can also use an organizational account (Microsoft 365 domain).

2. Create a Azure DevOps organization, and select a URL for your account. You’ll use this URL to connect from your development computer when you’re configuring source control in Visual Studio. The following image is an example of the account URL.

   ![Account URL](https://myteam.visualstudio.com)

   When the account is created, you’re directed to your account main page where you’re prompted to create your first project.

3. Create a demo **Fleet Management** project.

   ![Create a demo Fleet Management project](https://robertbadewy.visualstudio.com/)

   Create a Azure DevOps team project
If you already have a Azure DevOps organization, go to your account using Internet Explorer. This topic uses .visualstudio.com as the example URL for illustration purposes.


2. Under Recent projects & teams, click New to create a new project.

   - **Visual Studio Online**
     - **Overview**
     - **Users**
     - **Rooms**
     - **Load test**

   **About Visual Studio Online**

   - **Features**
     - What does Visual Studio Online have to offer?
   - **Pricing**
     - Free for up to 5 users
   - **Learn**
     - Access online help for Visual Studio Online

   **Recent projects & teams**
   - **New**
   - **Browse**
   - **TFSOnlineProject1**
   - 10/30/2014

3. In the Project name field, enter Fleet Management, enter a Description, and then click Create project.

   **Create the recommended folder structure in your team project**

   If you have migrated your code from a previous version using the Lifecycle Services (LCS) automated code upgrade tool, the following folder structure is automatically created in your Azure DevOps team project.

   - Trunk
   - Main
     - Metadata
     - Projects

   The Metadata folder contains your source XML files organized by packages and models and the Projects folder contains Visual Studio projects. If you are not migrating code and are starting from scratch, create a similar folder structure on the server in your team project before you start development.

   **Configure Visual Studio to connect to your team project**

   1. Start Visual Studio. If you are logged into the machine as an administrator, then you must start Visual Studio as an administrator.

   2. Click Tools > Options > Source Control > Plug-in Selection.

   3. In the Current source control plug-in field, select Visual Studio Team Foundation Server.

   4. Select Team > Connect to Team Foundation Server.

   5. In Team Explorer, click Select Team Projects.

   6. In the Select a Team Foundation Server drop-down list, select the Azure DevOps organization that hosts the Fleet Management project, or click Servers if it isn’t in the menu.

      a. When the Add/Remove Team Foundation Server dialog opens, click Add.
      b. Enter the URL of your Azure DevOps organization.
      c. Click OK.
      d. If prompted, enter your Microsoft Account username and password.
7. Select the Fleet Management check box under Team projects, and then click Connect.

Map your Azure DevOps project to your local model store and projects folder

Your model store root folder contains source files of all packages and models that are part of your application. During deployment, you'll probably use source files from more than one model across more than one package. We recommend that you map your model store root folder to the Azure DevOps team project metadata folder.

1. In Visual Studio Team Explorer, connect to the team project as described earlier in this document.

2. Open Source Control Explorer from Team Explorer.

3. Map the Metadata folder of your team project to the root folder of the model store on your local drive (Typically K:\AOSService\PackagesLocalDirectory), an example is shown in the image below.

   **NOTE**
   Your model store may be located under I:\AosService\PackagesLocalDirectory or another drive, depending on your machine configuration.

4. Click Map, and on the next dialog, click No.

5. Similarly, map the /Trunk/Main/Projects server folder to the local projects folder that will hold your Visual Studio solution and project files.
Scenario 1: Open the fleet management solution and add it to Azure DevOps source control

This section describes the steps needed to add a solution to Azure DevOps source control. This scenario is relevant when you have started development on a new model and you are adding it to source control for the first time. For code migration scenarios or in the case you are synchronizing new models that have been created by another developer, refer to scenario 2 below.

Open the FleetManagement solution

1. On the **File** menu, point to **Open**, and then click **Project/Solution**.
2. Browse to the desktop and open the **FleetManagement** folder.
3. Select the solution file named **FleetManagement**. The file type listed is Microsoft Visual Studio Solution. If the solution file is not on your computer, the steps to create it are listed in Tutorial: Create a Fleet Management solution file out of the Fleet Management models in the AOT.
4. Click **Open**. Loading the solution may take some time.

Add the FleetManagement solution to source control

1. In **Solution Explorer**, right-click the Fleet Management solution, and select **Add Solution to Source Control**.
2. On the next dialog, select **Team Foundation Version Control**, and then click **Next**.
3. In the **Team Project Location**, select **Projects**, as shown in this image.

**NOTE**

If you have already mapped the server Projects folder to a local folder that contains the FleetManagement solution, steps 2 and 3 are omitted.
4. Click OK.

5. Go to Team Explorer > Pending changes, and then click Check-in to check-in your solution and its model element to the Azure DevOps source control.

Add the model descriptor file to source control

All Visual Studio projects belong to models. Models are source code distribution and deployment units that are typically larger in scope than a Visual Studio project. In the previous section, you added element files of the fleet management solution to source control. Because this was the first time that you added elements of the Fleet Management models to source control, you'll also need to check-in the model descriptor file.

1. In Visual Studio, in Team Explorer, open Source Control Explorer.

2. Right-click the metadata folder (for example, Trunk\Main\Metadata) and click Add Items to Folder....

3. Select your model descriptor file. The model descriptor file is the XML file manifest of your model. It's located in the Descriptor folder of the package that the model belongs to. The following image shows an example of where the model descriptor file of the Fleet Management model exists (c:\packages\FleetManagement\Descriptor\FleetManagement.xml).

NOTE

Your model store may be located under K:\AosService\PackagesLocalDirectory or c:\AosService\PackagesLocalDirectory or another drive, depending on your machine configuration.
4. Click **Finish**.

**NOTE**
Because your solution contained elements from two models, you'll need to add an additional model descriptor file to source control:

K:\AOSService\PackagesLocalDirectory\FleetManagementExtension\Descriptor\FleetManagementExtension.xml

5. Check-in your pending items. Your item is now ready for development of the fleet management application using a state-of-the-art, cloud-based source control system, and many other application lifecycle features of Azure DevOps.

**Experiment with source control**
In this section, you'll make minor changes to the FMRental table and compare your changes with the latest version in your source code repository.

1. In **Solution Explorer**, select **Fleet Management Migrated > Data model > Tables > FMRental**.

2. Double-click **FMRental** to open the designer.

3. Right-click the **Fields** node, and then click **New > Integer**.
4. Right-click Methods, and add a new method.

5. In the X++ code editor, enter a comment in the new method.

6. Enter a comment in any existing method.

7. Save the FMRental.

8. In Team Explorer, right-click FMRental.xml, and select Compare with Latest Version.

9. Browse through the differences in the comparison (Diff) window.

10. In Solution Explorer, right-click on the FMRental table, and select Source control > Undo > Pending Changes to revert your changes.
11. Confirm the undo on the next dialog and close the diff window.

Scenario 2: Synchronize models from source control

In this section, you will synchronize existing models and model elements from your Azure DevOps project. Synchronization is relevant in the following cases: 1) You have migrated your code from a previous version via LCS, or 2) another developer has checked-in a new model or new model elements and you would like to synchronize them to your development environment.

1. In Source Control Explorer, right-click on Metadata and select Get Latest Version. Getting the latest version will synchronize you local packages folder with the latest code.

2. Alternatively you can use the Advanced menu to synchronize specific build version or change sets.

3. Once synchronization is complete, and if the synchronization leads to synchronizing new models to your environment, you need to refresh your metadata from Visual Studio.

4. Go to Dynamics 365 > Model Management > Refresh models.

Organize TODO tasks in a project

This section describes how you can create a Visual Studio project out of tasks (TODO comments) embedded in your X++ code.

1. In Solution Explorer, select Fleet Management Migrated > Code > Classes > FMDataHelper, and then double-click FMDataHelper. The X++ code editor opens.

2. Enter a TODO comment (/\TODO: my comment) inside any method.

   /// <summary>
   /// Create a number sequence reference for the default scope.
   /// </summary>
   /// <param name="_edt">The extended data type of the number sequence.</param>
   private static void InitializeNumberSequence(ExtendedTypeEd _edt, boolean resetWizardDefaults)
   {
     // TODO: This method needs to be modified to workaround bug 100
     NumberSequenceReference sequenceReference;
     NumberSequenceTable sequence;
     NumberSeqDataType dataTypeObject;
     NumberSequenceDatatype dataTypeRecord;
     NumberSeqScope scope;
   }

3. Open other Fleet Management classes or tables and add more TODO comments.
4. Rebuild the FleetManagement Migrated project.

5. Select View > Task List, to open the Visual Studio Task List window.

6. Select Comments from the drop-down list.

7. Select all TODO items, right-click, and select Add to new project.

8. Adding the items will open the New project dialog and enable you to create a new project that contains all of your TODOs.

9. You can save this project as a working project to manage your TODO list.

10. When you’re finished, undo all of your pending changes in Team Explorer.
11. Click File > Close Solution, to close the FleetManagement solution.

Use metadata search and navigation tools to find elements and create projects

This section demonstrates how you can perform meta-data based searches throughout your application.

Use the Metadata search window

1. Click Dynamics 365 > Metadata search.

2. In the Metadata search window, in the Search field, enter the following text to find all of the table insert methods in the Application Suite model that contain a cross-company query.

   `type:table,method name:insert code:"crosscompany" model:"Application Suite"`

3. Wait for the search to complete. It may take a while.

4. Double-click a result in the list. The code editor will open and place the cursor at the line that matches your search criteria.

5. Select several elements in the results list by holding down the Ctrl key for multiple selections, right-click, and then select Add to new project. Adding the elements will let you to create a new solution and
Try other search examples
You don’t need to wait for the search to complete before you interact with search results. You can double-click results at any time to view the metadata or source code that matches your search criteria. The following are some suggested search examples:

- Find vertical tab controls defined in view mode and autowidth mode in the model Application Suite.
  
  ```
  type:form,formtabcontrol property:arrangeMethod=Vertical,ViewEditMode=view,WidthMode=Auto
  model:"Application Suite"
  ```

- Find all grid controls in forms that aren't editable and with the property heightmode = column.
  
  ```
  type:form,formgridcontrol:allowedit=no,heightmode=column
  ```

- Find all SimpleListDetail forms in the Application Suite model.
  
  ```
  type:formdesign property:style=simplelistdetail model:"Application Suite"
  ```

- Find all tables that have an index field name that contains the keyword xpNum.
  
  ```
  type:table,tableindexfield anem: xpNum*
  ```

- Use the search bar drop-down menu to access previous searches.

Navigate to related elements
This section highlights a feature that enables you to move from one element to a related element without having to find the related elements in Application Explorer or Solution Explorer.

1. Open Application Explorer, and switch the view to Model View.

2. Under the Fleet Management model, click User Interface > Menu items > Display Menu Items > FMCustomer.
3. Right-click FMCustomer, and then select Open designer.

4. In the FMCustomer menu item designer, right-click the root node, and then select Go to Form FMCustomer.

The FMCustomer form designer will open.

5. In the designer of the FMCustomer form, expand Data sources, right-click FMCustomer, and then select Go to Table FMCustomer
Use Application Explorer to create a project from a model

You can use Application Explorer to search for all or some elements of a model and create a project out of the search results.

1. Make sure the option to organize a project by element type is on. Go to **Dynamics 365 > Options > Projects**.

2. Go to Application Explorer and search for elements in the desired model. For example, enter **model:“fleet management”** and click **Enter**.
3. When the search is complete, right-click the AOT root node and select Add search results to new project.

4. Specify your project properties in the new project dialog and click OK to create the project.

**TIP**

To create a project from search results, you can add any type, name, or other filters to your search as long as all results are in the same model. For example: `model:"Fleet Management" type:Table name:^FM` will return all tables in the Fleet Management model with a name starting with the letters FM.
You can automate the process of building X++ code and creating deployable packages on any build agent that run on Microsoft Windows. These agents include Microsoft-hosted agents. This approach helps you avoid the setup, maintenance, and cost of deploying build virtual machines (VMs). It also lets you reuse the existing setup of build agents to run other .NET build automation.

**NOTE**

This feature is limited to compilation and packaging. There is no support for X++ unit testing (SysTest), database synchronization, or other features that require the runtime (Application Object Server [AOS]) or its components.

**Prerequisites for building X++ code**

**Build projects**

To use .NET tools for building X++ in Azure DevOps, the Microsoft Build Engine (MSBuild) and custom X++ targets are used. Your X++ source code repository must contain an X++ project for each package that you have to build. You can optionally use a solution file to group the projects, including C# project dependencies, and provide an explicit build order. If the repository doesn't already contain a project, you can create a project in Visual Studio.

**NOTE**

When you use an existing X++ project (mrnproj), make sure that you created it, or opened and saved it, by using Visual Studio tools on Platform update 27 or later.

Although a package can contain multiple models, it must always be built in its entirety. Therefore, only one project for just one of the models is required to build the whole package. Additionally, although the project doesn't have to contain any objects, it can contain them.

**NuGet packages**

To build X++ code, the basic developer tools such as the X++ compiler (xppc.exe) are required. Additionally, any referenced packages, such as the Application Platform or Application Suite, must be available in a compiled format. To enable this process, the Shared asset library in Microsoft Dynamics Lifecycle Services (LCS) provides NuGet packages that are required to do an X++ build.

The following packages can be downloaded from the Shared asset library:

- **Microsoft.Dynamics.AX.Platform.CompilerPackage** – This package contains the X++ compiler and related tools that are required to do a build.
- **Microsoft.Dynamics.AX.Platform.DevALM.BuildXpp** – This package contains the compiled X++ code for the Application Platform and related modules. This code is optimized for building.
- **Microsoft.Dynamics.AX.Application.DevALM.BuildXpp** – This package contains the compiled X++ code for the Application and related modules. This code is optimized for building.

Starting from version 10.0.18, the Application Suite package has been split into two packages and there is an additional package to download from the Shared asset library.
Creating the pipeline

- **Microsoft.Dynamics.AX.ApplicationSuite.DevALM.BuildXpp** - This package contains the compiled X++ code for the Application Suite module. This code is optimized for building.

Download these packages from LCS, and add them to an Azure Artifacts feed in the Azure DevOps organization where the builds will run. For more information about how to create an Azure Artifacts feed and add NuGet packages, see these topics:

- Get started with NuGet packages in Azure DevOps Services and TFS
- Create a feed
- Create and publish your own NuGet package

**NOTE**

Free Azure DevOps organizations have limited storage for Azure Artifacts. Consider deleting old and unused versions to free up storage capacity. For more information, see Sign up for Azure Artifacts.

To identify which packages should be used during the build, and where they can be found, you must provide a nuget.config file and a packages.config file during the build. We recommend that you create these files and add them to the source control repository. The files can be stored anywhere in source control, because the paths of these files are explicit inputs for the NuGet command.

The nuget.config file provides NuGet with the source feed where the packages can be found. The packages.config file specifies the packages and their versions. To build against a newer version, you just have to update the versions in the packages.config file. For more information, including a sample nuget.config file, see Restore Package Management NuGet packages in Azure Pipelines.

The following example shows a packages.config file for the three main packages that are required for a typical X++ build. You must substitute the listed version with the actual versions of your NuGet packages.

- For version 10.0.17 or earlier, use the following packages.config layout:

  ```xml
  <?xml version="1.0" encoding="utf-8"?>
  <packages>
    <package id="Microsoft.Dynamics.AX.Platform.DevALM.BuildXpp" version="7.0.5644.16778" targetFramework="net40" />
    <package id="Microsoft.Dynamics.AX.Application.DevALM.BuildXpp" version="10.0.464.13" targetFramework="net40" />
    <package id="Microsoft.Dynamics.AX.Platform.CompilerPackage" version="7.0.5644.16778" targetFramework="net40" />
  </packages>
  ```

- For version 10.0.18 or later, use the following packages.config layout:

  ```xml
  <?xml version="1.0" encoding="utf-8"?>
  <packages>
    <package id="Microsoft.Dynamics.AX.Platform.DevALM.BuildXpp" version="7.0.5968.16973" targetFramework="net40" />
    <package id="Microsoft.Dynamics.AX.Application.DevALM.BuildXpp" version="10.0.793.16" targetFramework="net40" />
    <package id="Microsoft.Dynamics.AX.Platform.CompilerPackage" version="7.0.5968.16973" targetFramework="net40" />
  </packages>
  ```

Creating the pipeline
Azure DevOps provides pipelines that can be used to automate builds. There are two types of pipelines: YML and Classic. YML pipelines are available only when you use Git source control repositories. Classic pipelines must be used to build Team Foundation Version Control (TFVC) repositories. For more information, see Azure Pipelines.

This section describes the steps that are required in a pipeline to build X++ code. In the Dynamics365-Xpp-Samples-Tools GitHub repository, you can find a sample pipeline that can be imported into an existing Azure DevOps project.

Creating a basic build pipeline
A basic pipeline for compiling X++ requires only two steps:

1. Install the NuGet packages.
2. Build the solution or projects.

To simplify use of the extracted NuGet packages, consider using the **NuGet install** option and specifying the **-ExcludeVersion** NuGet command-line option. In that way, the extracted package paths can be used in the build, regardless of the version of the packages. Use the **NuGet Installer** task, and set the **Installation type** field to **Install**. Finally, specify the path of the packages.config and nuget.config files that you created earlier.

The following example of NuGet arguments will prevent a subfolder from being created for the package versions and will extract the NuGet packages into `$(Pipeline.Workspace)\NuGets`.

```plaintext
-ExcludeVersion -OutputDirectory "$(Pipeline.Workspace)\NuGets"
```

To build X++ by using MSBuild, you must supply several arguments. In the pipeline step that builds the solution, you can specify these arguments in the **MSBuild Arguments** field.

<table>
<thead>
<tr>
<th>ARGUMENT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>/p:BuildTasksDirectory</td>
<td>The path of the extracted Compiler Tools NuGet package, including the subfolders in the DevAlm folder.</td>
</tr>
<tr>
<td>/p:MetadataDirectory</td>
<td>The path of the X++ source code.</td>
</tr>
<tr>
<td>/p:ReferenceFolder</td>
<td>A semicolon-separated list of paths that contain binaries of X++ packages that are referenced and required for compilation (for example, Application Platform and Application Suite). If the code that will be compiled has multiple packages that reference each other, the output directory should also be included here.</td>
</tr>
<tr>
<td>/p:ReferencePath</td>
<td>A semicolon-separated list of paths that contain any non-X++ binaries that are referenced and required for compilation. You should include the location of the extracted Compiler Tools NuGet package, because it might contain required references.</td>
</tr>
<tr>
<td>/p:OutputDirectory</td>
<td>The path where the compiler will create folders and binaries.</td>
</tr>
</tbody>
</table>

The following example of MSBuild arguments assumes that the NuGet packages are installed in `$(Pipeline.Workspace)\NuGets`, the X++ source code is in `$(Build.SourcesDirectory)\Metadata`, and the output of the compiler should go in `$(Build.BinariesDirectory)`.

- For version 10.0.17 or earlier, use the following arguments:
For version 10.0.18 or newer, use the following arguments:

```
/p:BuildTasksDirectory="$(Pipeline.Workspace)\NuGets\Microsoft.Dynamics.AX.Platform.CompilerPackage\DevAlm"
/p:MetadataDirectory="$(Build.SourcesDirectory)\Metadata"
/p:FrameworkDirectory="$(Pipeline.Workspace)\NuGets\Microsoft.Dynamics.AX.Platform.CompilerPackage"
/p:ReferenceFolder="$(Pipeline.Workspace)\NuGets\Microsoft.Dynamics.AX.Platform.DevALM.BuildXpp\ref\net40;$(Pipeline.Workspace)\NuGets\Microsoft.Dynamics.AX.Application.DevALM.BuildXpp\ref\net40;$(Build.SourcesDirectory)\Metadata;$(Build.BinariesDirectory)"
/p:ReferencePath="$(Pipeline.Workspace)\NuGets\Microsoft.Dynamics.AX.Platform.CompilerPackage"
/p:OutputDirectory="$(Build.BinariesDirectory)"
```

In the pipeline samples, variables for NuGet package names and paths are used to simplify these commands.

**Creating a full pipeline that includes packaging**

To be useful, the pipeline should include a versioning step and a packaging step. Before you can add these steps to a pipeline, the *Dynamics 365 Finance and Operations Tools* extension for Azure DevOps must be enabled and installed in the Azure DevOps organization. For information about how to install an extension for an organization, see the [Azure DevOps documentation](https://azure.microsoft.com/en-us/documentation/articles/devops/external-tools/).  

A full pipeline should consist of at least the following steps:

1. Install the NuGet packages.
2. Update the model versions.
3. Build the solution or projects.
4. Install NuGet 3.3.0 or earlier on the agent. (This step is required for the step that creates the deployable package.)
5. Create the deployable package.
6. Publish the deployable package artifact as the build output.

For the deployable package to be created, NuGet must be readily available on the build agent. Therefore, the **NuGet tool installer** task in Azure DevOps must be run before the step that creates the package.

**NOTE**

If your source code repository includes binary packages from third parties like ISVs, you have to explicitly add those to the packaging step. For more information, see [Create deployable packages in Azure Pipelines](https://azure.microsoft.com/en-us/documentation/articles/devops/pipelines/tasks/package/).  

**NOTE**

Because of semantic versioning features in NuGet version 3.4 and later, make sure that the task installs version 3.3.0 or earlier. Currently, deployable package generation doesn't support semantic versioning.

**Sample pipeline for X++ developers**
In the Dynamics365-Xpp-Samples-Tools GitHub repository, you can find a sample pipeline that can be imported into an existing Azure DevOps project.
Add license files to a deployable package in Azure Pipelines

When you update an environment by using a deployable package, a license might be required for independent software vendor (ISV) or partner X++ solutions. ISVs can create pipelines to automatically include licenses in release or build pipelines. Customers can create their own pipelines to combine the ISV deployable package and the license file.

This topic assumes a working knowledge of Azure Pipelines.

**NOTE**
Before you can add these steps to a pipeline, the Dynamics 365 Finance and Operations Tools extension for Azure DevOps must be enabled and installed in the Azure DevOps account. For more information about how to install an extension for an organization, see Install extensions.

Adding the task to a pipeline

to a deployable package. The following table describes the options that are available for this task.

<table>
<thead>
<tr>
<th>INPUT NAME</th>
<th>MANDATORY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search pattern for license files to add to the package</td>
<td>Yes</td>
<td>A list of license files on the build agent, or a search pattern for files on the build agent. To make the license files available on the build agent, you can add them to source control. Alternatively, they can be downloaded or generated in an earlier step of the pipeline. For more information, see File matching patterns reference.</td>
</tr>
<tr>
<td>Filename and path of the deployable package to update</td>
<td>Yes</td>
<td>The path and file name of an existing deployable package zip file that the license files should be added to.</td>
</tr>
</tbody>
</table>
Create deployable packages in Azure Pipelines

11/24/2021 • 3 minutes to read • Edit Online

If you want to deploy customizations to an environment, a deployable package is required in Microsoft Dynamics Lifecycle Services (LCS). You can create this package by using Azure Pipelines during a build or release process.

This topic assumes a working knowledge of Azure Pipelines.

NOTE

Before you can add these steps to a pipeline, the Dynamics 365 Finance and Operations Tools extension for Azure DevOps must be enabled and installed in the Azure DevOps account. For more information about how to install an extension for an organization, see Install extensions.

This Azure DevOps task requires that the X++ compiler tools be available on the agent. Either run this task on a build virtual machine (VM) agent, or use the Compiler Tools NuGet package. For more information about the NuGet package and how to install it in a pipeline, see Build automation using Microsoft-hosted agents and Azure Pipelines.

Add the task to a pipeline

To add the task to the build of your YML or Classic pipeline, search the task list for Create Deployable Package. The following table describes the options that are available for this task.

<table>
<thead>
<tr>
<th>INPUT NAME</th>
<th>MANDATORY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>X++ Tools Path</td>
<td>Yes</td>
<td>The path of the location of the X++ tools. This location is either the PackagesLocalDirectory\bin folder location on a build VM, or the location of the extracted NuGet file of the Compiler Tools NuGet package.</td>
</tr>
<tr>
<td>Location of the X++ binaries to package</td>
<td>Yes</td>
<td>The path that contains the folders that contain all the binaries for the X++ packages (modules) that you want to include in the deployable package. If this task is used in a build pipeline, this folder is typically the same as the compiler output folder.</td>
</tr>
<tr>
<td>INPUT NAME</td>
<td>MANDATORY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Search pattern for binaries to package</td>
<td>Yes</td>
<td>Provide a name matching pattern for X++ package (module) names inside the path that is specified in the Location of the X++ binaries to package option. You can also specify a list of names instead of search patterns, or you can specify exclusion filters so that, for example, test packages aren't included. The search pattern looks for folders and validates that they contain a \bin sub-folder with X++ assemblies. For more information, see File matching patterns reference.</td>
</tr>
<tr>
<td>Filename and path for the deployable package</td>
<td>Yes</td>
<td>The path and file name of the deployable package. The output file is a zip file, and the file name typically includes version information to make the file easy to identify.</td>
</tr>
</tbody>
</table>

**NuGet dependency**

When this task is run on the build VM, NuGet is already available, and no action is required. However, when this task is run on hosted agents or other private agents, NuGet must be installed. In this case, Azure DevOps has the NuGet Tool Installer task that you can run before you run the task to create the package.

**NOTE**

Because of the introduction of semantic versioning in NuGet version 3.4 and later, you must install version 3.3.0 or earlier.

**Search for binaries to package**

Compared to the legacy packaging on the build virtual machine, the packaging task has to specify which modules to package and where to find them. In a standard pipeline, X++ modules under compilation are output in the binaries folder of the Azure DevOps agent. The packaging task by default will look in this folder for any X++ binaries. The search looks for folder names. The task will check inside these folders if there is a \bin subfolder with X++ assemblies.

**NOTE**

If your source control repositories includes third-party binaries such as ISV modules, the packaging step has specifically includes those binaries. See the examples section of this topic.

**Examples of search patterns**

The following example assumes the Location of the X++ binaries to package property is set to its default value of $(Build.BinariesDirectory), which is the location where the X++ compiler produces the binaries.

<table>
<thead>
<tr>
<th>SEARCH PATTERN</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>


<table>
<thead>
<tr>
<th>SEARCH PATTERN</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Find all X++ binaries in <code>${Build.BinariesDirectory}</code>. This is the default value.</td>
</tr>
<tr>
<td>*</td>
<td>Find all X++ binaries, exclude any module names that end in <code>Tests</code>.</td>
</tr>
<tr>
<td>*</td>
<td>Include all X++ binaries in <code>${Build.BinariesDirectory}</code>, as well as a module named <code>MyBinaryPackage</code> in the sources directory (which is the mapped source control repository folder) inside the <code>Metadata</code> folder.</td>
</tr>
<tr>
<td>*</td>
<td>Include all X++ binaries in <code>${Build.BinariesDirectory}</code>, exclude any modules where the names end in <code>Tests</code>, and include two modules named <code>MyISV1</code> and <code>MyISV2</code> in the sources directory (which is the mapped source control repository folder) inside the <code>Metadata</code> folder.</td>
</tr>
</tbody>
</table>
During build automation, X++ model versions can be updated so that they match or are linked to the build number of the pipeline. These updates make it easier for customers to identify the version of the X++ packages that they are running. They also let developers track versions back to the build pipeline and the version of the source code files.

This topic assumes a working knowledge of Azure Pipelines.

NOTE

Before you can add these steps to a pipeline, the Dynamics 365 Finance and Operations Tools extension for Azure DevOps must be enabled and installed in the Azure DevOps account. For more information about how to install an extension for an organization, see Install extensions.

Add the task to a pipeline

To add the task to the build of your YML or Classic pipeline, search the task list for Update Model Version. The following table describes the options that are available for this task.

<table>
<thead>
<tr>
<th>INPUT NAME</th>
<th>MANDATORY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>X++ Source Location</td>
<td>Yes</td>
<td>The path of a parent folder that contains X++ source code. The Descriptor Search Pattern option will be run in this path and subfolders. The default value, $(Build.SourcesDirectory), points to the root of the source code repository.</td>
</tr>
<tr>
<td>Descriptor Search Pattern</td>
<td>Yes</td>
<td>Provide a file matching pattern to find the descriptor files inside the path that is specified in the X++ Source Location option. You can also specify a list of full paths of descriptor files instead of search patterns. For more information, see File matching patterns reference.</td>
</tr>
<tr>
<td>Lowest Layer to Update</td>
<td>Yes</td>
<td>When using search patterns, this task may find descriptors for ISV or partner code in your source control. Those descriptor versions should likely not be updated. This option allows you to define an additional filter to define the lowest layer that the task will update.</td>
</tr>
<tr>
<td>INPUT NAME</td>
<td>MANDATORY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Version number in format #.#.#.# to update</td>
<td>Yes</td>
<td>The version number to write into the descriptors. Note that the format must consist of four digits that are separated by periods (.). By using the build ID or build number, you allow for traceability. Note that the default build number isn’t in the correct format. You can change the format in the classic editor, under Options &gt; Build number format. You can also change it by adding a name: tag at the top of the YML file. An example of a valid build number that uses the year, month, and date, and also the build count revision number, is $(Date:yy.MM.dd)(Rev:.r)$. For more information, see Configure run or build numbers.</td>
</tr>
</tbody>
</table>
You can automate the download of assets from the Asset library in Microsoft Dynamics Lifecycle Services (LCS) by using the Deploy Lifecycle Services (LCS) Asset Download task in Azure DevOps.

This topic assumes that you have a working knowledge of Azure Pipelines.

**NOTE**

Before you can add these steps to a pipeline, the Dynamics 365 Finance and Operations Tools extension for Azure DevOps must be enabled and installed in the Azure DevOps account. For more information about how to install an extension for an organization, see Install extensions.

## Add the task to a pipeline

To add the task to the build of your YML or Classic pipeline, search the task list for Dynamics Lifecycle Services (LCS) Asset Download.

The following table describes the options that are available for this task.

<table>
<thead>
<tr>
<th>INPUT NAME</th>
<th>MANDATORY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCS Connection</td>
<td>Yes</td>
<td>Select or create a service connection to LCS. For more information, see Create an LCS connection in Azure Pipelines.</td>
</tr>
<tr>
<td>LCS Project ID</td>
<td>Yes</td>
<td>Enter the ID of the project in LCS that contains both the asset to deploy and the target environment. You can find the project ID at the end of the URL of your project's dashboard.</td>
</tr>
<tr>
<td>Path to download to</td>
<td>Yes</td>
<td>Enter the path to download the asset to.</td>
</tr>
</tbody>
</table>
Select the type of search pattern that should be used to find the asset in the Asset library in LCS. Depending on the value that you select, the following options are available:

- **Asset ID (guid)** – If you select this value, in the LCS File Asset Id(s) field, enter the asset ID or a semicolon-separated list of asset IDs. Asset IDs are globally unique identifiers (GUIDs).
- **Name** – If you select this value, in the LCS File Asset Type field, select the asset type. Then, in the LCS File Asset Name field, specify the name to search for. You can use an asterisk (*) as a wildcard character in the name. For example, you might enter MyPackage*.

After a successful download, an output variable can be used to capture a list of the file paths. If there are multiple files, a semicolon-separated list of file paths is assigned to the output variable. For more information about output variables in Azure DevOps, see Use output variables from tasks.
You can automate the upload of assets to the Asset library in Microsoft Dynamics Lifecycle Services (LCS) by using the **Deploy Lifecycle Services (LCS) Asset Upload** task in Azure DevOps. This task is available only in **Releases** pipelines.

This topic assumes you have a working knowledge of Azure Pipelines.

**NOTE**

Before you can add these steps to a pipeline, the Dynamics 365 Finance and Operations Tools extension for Azure DevOps must be enabled and installed in the Azure DevOps account. For more information about how to install an extension for an organization, see Install extensions.

### Add the task to a pipeline

To add the task to the build of your YML or Classic pipeline, search the task list for **Dynamics Lifecycle Services (LCS) Asset Upload**.

The following table describes the options that are available for this task.

<table>
<thead>
<tr>
<th>INPUT NAME</th>
<th>MANDATORY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCS Connection</td>
<td>Yes</td>
<td>Select or create a service connection to LCS. For more information, see Create an LCS connection in Azure Pipelines.</td>
</tr>
<tr>
<td>LCS Project ID</td>
<td>Yes</td>
<td>Enter the ID of the project in LCS that contains both the asset to deploy and the target environment. You can find the project ID at the end of the URL of your project’s dashboard.</td>
</tr>
<tr>
<td>Type of asset</td>
<td>Yes</td>
<td>Select the type of asset to upload.</td>
</tr>
<tr>
<td>File to upload</td>
<td>Yes</td>
<td>Enter the path of the file that you want to upload into the Asset library in LCS.</td>
</tr>
<tr>
<td>LCS Asset Name</td>
<td>No</td>
<td>Enter the name to show for the asset in the Asset library. If you don't enter a name here, the file name will be used.</td>
</tr>
<tr>
<td>LCS Description</td>
<td>No</td>
<td>Enter the description to show for the asset in the asset details.</td>
</tr>
<tr>
<td>Wait for Validation</td>
<td>No</td>
<td>For asset types that require validation, use this check box to instruct the task to wait until validation of the asset has either succeeded or failed.</td>
</tr>
</tbody>
</table>
You can use the **Deploy Lifecycle Services (LCS) Asset Deployment** task in Microsoft Azure DevOps to automate the deployment of assets that are stored in the Asset library in Microsoft Dynamics Lifecycle Services (LCS) to specific environments. However, this task has the following limitations that you should consider:

- The task is available only in **Releases** pipelines.
- The deployment of software deployable packages to production environments can't be automated.
- Software deployable packages can't be deployed to build environments.
- Software deployable packages can't be deployed to local business data (LBD) environments on-premises.

This topic assumes that you have a working knowledge of **Azure Pipelines**.

### Add the task to a pipeline

To add the task to the build of your YML or Classic pipeline, search the task list for **Dynamics Lifecycle Services (LCS) Asset Deployment**. If your target environments include self-service environments, be sure to select task version 1.* or later.

The following table describes the options that are available for this task.

<table>
<thead>
<tr>
<th>INPUT NAME</th>
<th>MANDATORY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCS Connection</td>
<td>Yes</td>
<td>Select or create a service connection to LCS. For more information, see <a href="https://docs.microsoft.com/en-us/azure/devops/pipelines/tasks/dynamic-lifecycle-service/create-lcs-connection?view=azure-devops">Create an LCS connection in Azure Pipelines</a>.</td>
</tr>
<tr>
<td>LCS Project ID</td>
<td>Yes</td>
<td>Enter the ID of the project in LCS that contains both the asset to deploy and the target environment. You can find the project ID at the end of the URL of your project's dashboard.</td>
</tr>
<tr>
<td>LCS Environment ID</td>
<td>Yes</td>
<td>Enter the ID of the target environment. The environment ID is a globally unique identifier (GUID) that you can find on the environment's details page, under Environment Details &gt; Environment ID.</td>
</tr>
<tr>
<td>INPUT NAME</td>
<td>MANDATORY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LCS File Asset ID</td>
<td>Yes</td>
<td>Enter the asset ID of the software deployable package to deploy. The asset ID is a GUID that you can find in the Asset library. Select the row of the asset that you want to deploy, and then, under Additional Details, look in the Asset ID field. Typically, this ID comes dynamically from other pipeline steps, such as the Dynamics Lifecycle Services (LCS) Asset Upload task.</td>
</tr>
<tr>
<td>Name for the update</td>
<td>Yes</td>
<td>Enter the name that is shown for the update in the environment history in LCS.</td>
</tr>
<tr>
<td>Wait for Completion</td>
<td>Cleared (No)</td>
<td>Use this check box to instruct the task to wait until the deployment of the asset has either succeeded or failed. If it's cleared (No), the task will only start the deployment. If the task is instructed to wait, a pipeline time-out might occur during long-running deployments. For more information about time-out options, see Timeouts.</td>
</tr>
</tbody>
</table>
Create an LCS connection in Azure Pipelines

11/24/2021 • 3 minutes to read • Edit Online

The Dynamics 365 Finance and Operations Tools extension for Microsoft Azure DevOps has several pipeline tasks that let you perform actions in Microsoft Dynamics Lifecycle Services (LCS). For example, you can upload assets, download assets, and service an environment. For the connection with LCS to work, you must set up a new service connection in Azure DevOps. This service connection provides the authentication details that are required to connect to LCS. For more information about service connections in Azure DevOps, see Service connections.

This topic assumes that you have a working knowledge of Azure Pipelines.

**NOTE**

Before you can add these steps to a pipeline, the Dynamics 365 Finance and Operations Tools extension for Azure DevOps must be enabled and installed in the Azure DevOps account. For more information about how to install an extension for an organization, see Install extensions.

Prerequisites

You must have the credentials for a user who has access to one or more LCS projects that you want to interact with from Azure DevOps. Make sure that this user has successfully signed in to LCS before, and has opened the dashboard for the projects that you want to interact with.

**NOTE**

LCS doesn’t support service-to-service authentication. Therefore, only regular user credentials (that is, a user name and password) can be used. Because the pipelines don’t run interactively, multifactor authentication must not be set up for the account that you use. We recommend that you set up a separate user account that has limited access and strong credentials that can regularly be rotated for security purposes.

To enable direct connections from Azure DevOps to LCS on a user’s behalf, you must register an application in your Azure Active Directory (Azure AD).

1. Follow the instructions in Quickstart: Register an application with the Microsoft identity platform, and add a new redirect URI:
   a. Select Public client/native (mobile & desktop).
   b. Enter any valid URI, such as http://localhost.

2. Add permissions to the application registration to access the LCS web APIs. Follow the instructions in Add permissions to access your web API. When you request the API permissions, select APIs my organization uses, and search for Dynamics Lifecycle services.

3. Make sure that the account that you will use has given consent for the application registration in Azure AD. Follow the instructions in Configure the way end-users consent to an application in Azure Active Directory. You can either enable a specific user or grant admin consent for the whole tenant.

4. Configure the registration as a public client application.
   a. In the Azure portal, select your app in App registrations, and then select Authentication.
   b. In Advanced settings > Allow public client flows > Enable the following mobile and
Create the Dynamics Lifecycle Services service connection

You can create a new service connection either directly from a pipeline task or from your project’s settings page. For more information about how to create service connections, see [Create a service connection](#). In the dialog box for the **Dynamics Lifecycle Services** service connection, provide the following information:

- **Authentication Endpoint** – The default value works for all Azure AD tenants in the Azure cloud. If your Azure AD is in a national cloud, see [National clouds](#) to find the correct authentication endpoint.
- **Lifecycle Services API Endpoint** – Provide the endpoint.
- **Username** – Provide the email alias for the user credentials.
- **Password** – Provide the password for the user credentials.
- **Application (Client) ID** – Provide the ID that was previously created for your application registration in Azure AD.
- **Service connection name** – Provide a meaningful name for the connection. This name will appear in the connection field for pipeline tasks that require an LCS connection.
- **Description** – Provide an optional description of this connection.
Update the hosted Azure Pipeline for new NuGet packages

11/24/2021 • 2 minutes to read • Edit Online

NOTE
This topic applies to pipelines that were set up for versions 10.0.17 or earlier. This does not apply to the legacy build pipeline that uses the build virtual machine.

Platform updates for version 10.0.18 introduce a new NuGet package. The new package is a result of a package split for the Application Build Reference code. As a result, you have to make changes to pipelines created for 10.0.17 or earlier versions.

Add the new package to packages.config list

The packages.config file used for your build already includes three packages:

```xml
<?xml version="1.0" encoding="utf-8"?>
<packages>
  <package id="Microsoft.Dynamics.AX.Platform.DevALM.BuildXpp" version="7.0.5934.35741" targetFramework="net40" />
  <package id="Microsoft.Dynamics.AX.Application.DevALM.BuildXpp" version="10.0.761.10019" targetFramework="net40" />
  <package id="Microsoft.Dynamics.AX.Platform.CompilerPackage" version="7.0.5934.35741" targetFramework="net40" />
</packages>
```

You need to add a fourth package to the list, Microsoft.Dynamics.AX.ApplicationSuite.DevALM.BuildXpp. The resulting packages.config file should look like the following example, replacing the version number with the version number that your pipeline uses.

```xml
<?xml version="1.0" encoding="utf-8"?>
<packages>
  <package id="Microsoft.Dynamics.AX.Platform.DevALM.BuildXpp" version="7.0.5968.16973" targetFramework="net40" />
  <package id="Microsoft.Dynamics.AX.Application.DevALM.BuildXpp" version="10.0.793.16" targetFramework="net40" />
  <package id="Microsoft.Dynamics.AX.ApplicationSuite.DevALM.BuildXpp" version="10.0.793.16" targetFramework="net40" />
  <package id="Microsoft.Dynamics.AX.Platform.CompilerPackage" version="7.0.5968.16973" targetFramework="net40" />
</packages>
```

Add a pipeline variable

The pipeline uses variables to simplify and centralize parameters used in the pipeline tasks. There are already variables for each of the NuGet package names. To add a variable for the name of the new NuGet package, do the following:

1. On the Variables tab of the pipeline, select the Add link at the bottom of the list of variables.
2. In the Name column, type: AppSuitePackage.

Update the Build solution step

In the Build solution step in the pipeline, the path and names to all the NuGet packages are supplied as command-line parameters to MSBuild. To add the new NuGet package to the semi-colon separated list of ReferenceFolder paths, do either of the following:

- If you used the existing template without modifying it, the new MSBuild Arguments will be:
  ```
  /p:BuildTasksDirectory="$(NugetsPath)\$(ToolsPackage)\DevAlm" /p:MetadataDirectory="$(MetadataPath)" /p:FrameworkDirectory="$(NugetsPath)\$(ToolsPackage)\" /p:ReferenceFolder="$(NugetsPath)\$(ToolsPackage)\$\$(ToolsPackage)\$(AppPackage)\ref\net40;$(MetadataPath);$(Build.BinariesDirectory);$(NugetsPath)\$(ToolsPackage)\ref\net40" /p:ReferencePath="$(NugetsPath)\$(ToolsPackage)" /p:OutputDirectory="$(Build.BinariesDirectory)"
  ```
- If you’ve modified the arguments list, find the ReferenceFolder property argument and add $(NugetsPath)\$(AppSuitePackage)\ref\net40 to the semi-colon separated list. Add a semi-colon to separate this new entry from other paths in the list.

Use the updated templates on GitHub as an alternative

As an alternative to making these changes, or as a way to verify your changes, review the updated templates in the Dynamics365-Xpp-Samples-Tools GitHub repository.
Update a legacy pipeline in Azure Pipelines

11/24/2021 • 2 minutes to read • Edit Online

NOTE
This documentation does not apply to the new build pipeline, even if you run it on the build virtual machine.

Even though Visual Studio 2017 is available on virtual machines deployed with version 10.0.13 or later, the build scripts needed to support unit testing using VSTest require version 10.0.15.

An Azure Pipelines pipeline explicitly specifies the versions of Visual Studio tools such as MSBuild and VS Test. To continue supporting newer versions of these products, new virtual machines will be deployed with newer versions of Visual Studio pre-installed. Any existing pipelines need to be manually updated to use the newer version.

Determine if your pipeline needs to be updated

Build and development virtual machines deployed with version 10.0.13 include Visual Studio 2017. Support for Visual Studio 2015 will be deprecated in the April 2021 release. If your build virtual machine does not include Visual Studio 2017 you must plan to deploy a new build virtual machine with version 10.0.13 or later, or consider using the new hosted build pipeline.

If your build virtual machine has Visual Studio 2017 installed, you can use the newer versions of MSBuild and VS Test. If your pipeline was created by a virtual machine deployment prior to version 10.0.13, you will have to manually update the pipeline.

Finally, you can check your build pipeline on the Build the solution step. The MSBuild Version property indicates the version of Visual Studio in use.

<table>
<thead>
<tr>
<th>MSBUILD VERSION</th>
<th>VISUAL STUDIO VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSBuild 14.0</td>
<td>Visual Studio 2015</td>
</tr>
<tr>
<td>MSBuild 15.0</td>
<td>Visual Studio 2017</td>
</tr>
</tbody>
</table>

Updating the Azure Pipelines pipeline

NOTE
To update an Azure Pipelines pipeline to use Visual Studio 2017, ensure your build virtual machine has been updated to version 10.0.15 or newer.

The following four properties, in three tasks in the pipeline, need to be updated.

<table>
<thead>
<tr>
<th>TASK NAME</th>
<th>TASK PROPERTY</th>
<th>OLD VALUE</th>
<th>NEW VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build the solution</td>
<td>MSBuild Version</td>
<td>MSBuild 14.0</td>
<td>MSBuild 15.0</td>
</tr>
<tr>
<td>Database Sync</td>
<td>MSBuild Version</td>
<td>MSBuild 14.0</td>
<td>MSBuild 15.0</td>
</tr>
<tr>
<td>TASK NAME</td>
<td>TASK PROPERTY</td>
<td>OLD VALUE</td>
<td>NEW VALUE</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------</td>
<td>------------------------------------------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>Execute Tests</td>
<td>Other console options</td>
<td>/Platform:X64 /InIsolation /UseVsixExtensions:true</td>
<td>/Platform:X64 /InIsolation /TestAdapterPath:&quot;$(VsixExtensionFolder)&quot;</td>
</tr>
</tbody>
</table>
This tutorial walks you through an end-to-end scenario that the Fleet Management sample application is designed to support.

In this tutorial, you’ll take a tour of the Fleet Management sample. The overviews in this tutorial provide some background knowledge and contextual info. You’ll walk through an end-to-end scenario that this sample application is designed to support. This is information that you should have before proceeding to other tutorials.

Prerequisite

- You must first be provisioned as an end user before you start this tutorial.
- This tutorial mainly explores the FleetManagement Migrated project and the application that it builds.

Installing the demo data

To work with the sample, you must install the provided demo data.

1. In the virtual machine (VM), open Internet Explorer and navigate to the application's base URL.
2. Sign in.
3. On the dashboard, open the navigation pane and go to Fleet Management > Setup > Fleet setup.
4. On the Data setup tab, select Create.
5. If you’re prompted to reload the demo data, click Yes.

6. When the data is finished loading, select Close.

Use the Fleet Management application to rent a vehicle

Remember that you’re working with the migrated app in this section. The forms that you see are directly ported from the Microsoft Dynamics AX 2012 version of the sample. Although they have been modified and restyled, they have not been reimagined.

1. Open Internet Explorer, and sign into the Finance and Operations application.

2. To return to the Dashboard, select the product name in the top-left corner of the page.

The dashboard is the main working hub. You can see the various tiles, organized into sections, which lead to parts of the application. The dashboard is designed for horizontal scrolling, which is an optimization for working well on modern devices. The button to the right of the dashboard shows the navigation bar.

3. From the Dashboard, open the navigation bar and go to Fleet Management > Customers > Customer.
4. To switch to the **Details** view, select a value in the **First Name** column. This view shows detailed information for a single customer.
5. Click **Show list** to show the navigation list.

6. Click the various customer names in the navigation list in the side pane, and watch as the detailed information about each customer changes.

7. Select the customer **Eduardo Cobo**. You'll notice the charts update to indicate Eduardo's previous rental preferences.
8. Hover over the pie slices to see the details. You'll notice that, in the past, Eduardo has often rented red SUVs. This might give the sales clerk a cue to look for available red SUVs the next time Eduardo makes a reservation. This is a simple example of proactively providing insights.

9. Add yourself as a customer.
   - On the Action Pane, click **New**.
     
     Fill in the form to add yourself as a customer. Make sure that you provide your name, a 16-digit number in the credit card field, and address information, at a minimum. **Note**: You don't have to take any action to save a new record.

10. Create a new rental.
    a. On the navigation bar, go to **Fleet management > Rentals > Rental**.
    b. In the **Rental** form, on the Action Pane, click **New**.
    c. In the **Vehicle** field, select a vehicle.
    d. In the **Customer** field, select your name.
    e. In the **To** field, pick an end date.
    f. In the **Start** field, enter **35,000**.
    g. In the **Pickup** field, enter **Full**.
    h. When you are done, click **Save**.

11. Start the rental period.
    a. On the Action Pane, click **Start rental**.
    b. In the dialog box, verify the values in the fields and click **OK**.
Use Fleet Management to run a workflow

1. Click the **Home** icon to return to the dashboard.

2. Find the **Reservation Management** tile and select it to open the Reservation Management workspace.

3. Click **Current rentals**.

4. On the **Rentals** form, click the ID of your rental.

5. On the Details view of the **Rentals** form, on the Action Pane, click **Complete rental**.

6. In the **New mileage** field, enter **40,000**, and then click **OK**.

7. Click the **Home** icon to return to the dashboard.

8. On the navigation bar, navigate to **Fleet management > Vehicles > Vehicle Maintenance**. In the **Vehicle Maintenance** form, the **Status** field shows that your rental is awaiting examination by the service department.

   ![Vehicle Maintenance Form](image)

<table>
<thead>
<tr>
<th>Vehcile maintenance</th>
<th>Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vehicles</td>
<td>Mileage</td>
</tr>
<tr>
<td>Datum_Four_1</td>
<td>40000</td>
</tr>
</tbody>
</table>

   **NOTE**
   You might need to wait up to two minutes for the batch framework to change the status of the vehicle. On the Action Pane, click **Refresh** periodically to update the view, until you see the status change. Keep in mind that a different person usually handles each step in a workflow; the brief delay introduced by the batch framework is not an issue in a real-world application.

9. Select the row that contains your rental. On the Action Pane, click **Workflow**, and then click **Examination complete**. You may need to refresh the page to get the full set of options under **Workflow**.

10. Enter a comment, and then click **Examination complete**.

11. You might again need to wait up to two minutes for the batch framework to process the change. On the Action Pane, click **Refresh** periodically, until you see the **Status** field change. Notice that the vehicle now has a status of **Awaiting Service**.

12. Optionally, you can continue to repeat these workflow steps to take the vehicle through the service and cleaning phases. After cleaning is completed, the final status is **Done**.

13. Click **Workflow**, and then click **View history**. The **Workflow history** form provides information about the vehicle workflow.

14. Click **Tracking details** to see the activities.
To view the setup behind the workflow

1. On the dashboard, navigate to Fleet Management > Setup > Workflow setup. The Workflow Setup page shows the list of workflows.

2. In the Workflow ID column, click the ID of your vehicle maintenance workflow.

3. Accept any prompts that ask you for permission to run code. After a short wait, the workflow editor opens. This step works on the one-box environment, but not in the cloud. You can view the workflow diagram in the workflow editor. The following illustration shows the workflow.
Create a new KPI definition

The web client enables users who have appropriate permissions to modify KPI definitions that have been modeled and deployed by developers. Users also have the ability to create new KPI definitions in the client. In this walkthrough, you create a new KPI definition in the client.


2. Notice the Total revenue KPI tile shown on the bottom left of the workspace. Click the Total Revenue KPI tile. Details of the total revenue KPI tile along with charts indicating top and bottom contributors to revenue will be shown on screen.

3. Next, you will define a new KPI to monitor the number of rentals.


5. Enter following values for the new KPI definition.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Number of Rentals</td>
</tr>
<tr>
<td>Measurement</td>
<td>FMAggregateMeasurements</td>
</tr>
</tbody>
</table>
### New KPI

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Number of Rentals</td>
</tr>
</tbody>
</table>

**Value**

92.00

**Measurement**

FMAggregateMeasureme... 

**Measure group**

FMRentalCharges

**Measure**

NoRentals

**Goal value**

30.00

6. Click **Save**.

**NOTE**

If the **Save** button isn’t visible in the **New KPI** dialog box, use a higher screen resolution so that you can see the entire dialog. You can see the KPI details page that contains details about the KPI that you created. You can make changes in the **Details** section. You will modify the default threshold values so that if the value is less than 90% of the goal, the KPI will show red and if the value is over 110% of the goal, the KPI will show green.

7. Click **Edit**.

8. Scroll to the right of the screen, and modify the values in the thresholds fields as follows.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bad if less than</td>
<td>90</td>
</tr>
<tr>
<td>Good if more than</td>
<td>110</td>
</tr>
</tbody>
</table>

9. In the application bar, click **Save**.
Launch an operational report

In this tutorial, you’ll launch an operational report that contains a list of customers who are currently renting vehicles.

1. Use the dashboard to open the Reservation management workspace.

2. On the right side of the page, under Reports click Customer list. Do not enter anything in the parameter for Customer group.
Customer list

Parameters

Customer group

Destination

Change

Screen

Run in the background

OK  Cancel

3. Click OK to close the dialog box. The report will be rendered and show the list of customers. The report may take a minute to render.

Secure access using the role-based security system

In this tutorial, you'll access the system as a user that has been assigned a different security role. This tutorial requires that you have created at least one additional end user.

1. On the dashboard, in the System administration section, click Users and then Users.
2. On the Action Pane, click New.
3. Enter the following field information.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>User ID</td>
<td>Eight character unique ID</td>
</tr>
<tr>
<td>User name</td>
<td>The first name of the user</td>
</tr>
<tr>
<td>Provider</td>
<td>urn:Federation:MicrosoftOnline</td>
</tr>
<tr>
<td>Email</td>
<td>Use an email alias that you can login with for testing.</td>
</tr>
<tr>
<td>Company</td>
<td>DAT</td>
</tr>
<tr>
<td>Enabled</td>
<td>Verify that this slider is set to Yes.</td>
</tr>
</tbody>
</table>
4. Click **Assign Roles**.

5. Select **Fleet management branch manager**, and then click **OK**.

6. Click the user name on the top right, and then click **Sign Out**. You’ll be redirected back to the sign-in page.

7. Sign in using the credentials for the user who you assigned the security role to in the steps above.

8. Notice that in the dashboard, this user can see only items that are related to the branch manager security role. Items that system administrators can see are now hidden.
9. Click **Sign out** to sign out of the session.
Fleet Management sample application

11/24/2021 • 2 minutes to read • Edit Online

This topic is an overview of the Fleet Management sample application.

The Fleet Management sample application has been provided to showcase development and foundation capabilities. Fleet Management represents a solution that an ISV might create for a car-rental agency. Fleet Management data includes vehicles which are available for renting, and customers who can rent and return these vehicles. Employees can also run a maintenance workflow on these vehicles.

For some tutorials, you will need to create the FleetManagement solution if it is not on your computer. The steps to create it are listed in Tutorial: Create a Fleet Management solution file out of the Fleet Management models in the AOT.

For some tutorials, you must download the Fleet Management tutorial code and other artifacts from https://github.com/Microsoft/FMLab.

Fleet Management is provided as a Visual Studio solution that demonstrates platform capabilities, such as:

- Forms
- Workflow
- Security
- Labels
- Resources
- Data
- Business Intelligence
- Extensions

The Fleet Management solution includes two separate projects: one for the base model and the other one for extensions to the base model. The project named FleetManagement Migrated demonstrates how a migrated application might appear after migrating code from Dynamics AX 2012. This version shows how forms that have been migrated from Microsoft Dynamics AX 2012 R3 work on a web client. These forms have been created using automated migration tools and some other manual migration steps in Visual Studio. These forms bind to X++ tables and use the X++ programming model. The project named FleetManagement Discounts (or FleetManagementExtension) demonstrates how to use extensions to customize an application. This project extends the Fleet Management sample by extending controls and tables, handling data events, and replacing business logic using a plug-in. The tutorials that accompany this article provide a more-detailed look at the Fleet Management sample. These include a Fleet Management tutorial, End-to-end scenario for the Fleet Management sample application, and a tutorial that walks through extensions, Customize model elements through extension.

Additional resources

Using the Fleet Management sample

Customize model elements using extensions

Develop and customize home page

Download the FMLab sample code

Create the FleetManagement solution
What are the development tools?

Application development is carried out in Visual Studio. The development tools support all of the development tasks, including debugging and local testing scenarios. Visual Studio is the exclusive integrated development environment (IDE) for development. All of your application development work will be performed with it. This section is an overview of the main features that are added to Visual Studio when the development tools are installed.

Visual Studio 2019 is supported beginning with platform updates for version 10.0.21 of Finance and Operations apps.

**Application Explorer**

In Visual Studio, the model store is represented by the Application Explorer. On the View menu, click **Application Explorer** to open it. Use the Application Explorer to browse and interact with the elements in the model store that define the applications. The following illustration shows the Application Explorer. For more information, see Application Explorer.
The project template

Even a simple application can have a large number of elements in its model. The Operations Project template has been added to Visual Studio to help you organize and manage the elements that you are working with for a model. You will use the project to design, build, and test model elements. It's common to have several projects within a single Visual Studio solution. The following illustration shows three projects in a Visual Studio solution. For more information, see Finance and Operations project type in Visual Studio.
Element designers

The Visual Studio tools contain designers for each kind of element in the application. You will use these designers when you create or modify elements. The following illustration shows the element designer for a form element. For more information, see Element designers.

Code editor

The X++ code is written in the code editor for Visual Studio. The standard features that a developer expects from the code editor are supported. For example, sections of code are collapsible. IntelliSense provides guidance as you write or modify code. For more information, see Code editor features.
Dynamics 365 menu

The tools add the Dynamics 365 menu to Visual Studio. Several tools that you will use during the development process are found here. For example, the tools for managing models are accessed from the menu.
This tutorial tours the Fleet Management solution in Visual Studio and introduces you to the development tools.

In this tutorial, you’ll take a tour of the Fleet Management solution in Visual Studio. You’ll see how a project is organized in the Visual Studio development environment. Much of what you’ll see uses standard Visual Studio features, plus you will notice that we’ve added some new customized features. Along the way we’ll point out some of these new and customized features and how they ease development. This tutorial will focus on:

- Visual Studio projects and their features.
- Files used in development.

Prerequisites

This tutorial requires you to access the environment using Remote Desktop and to be provisioned as an administrator for the instance.

Setup

1. Start Visual Studio using Run as an administrator.
2. On the File menu, point to Open and then click Project/Solution.
3. Browse to the Desktop, and then open the FleetManagement folder. If the solution file is not on your computer, the steps to create it are listed in Tutorial: Create a Fleet Management solution file out of the Fleet Management models in the AOT.
4. Select the solution file named FleetManagement. The file type listed is Microsoft Visual Studio Solution (SLN file).
   - The fleet management solution file is available on the downloadable VHD.
5. Click Open. The solution may take some time to load.

View the FleetManagement model

1. On the View menu, click Application Explorer. Application Explorer opens in Classic view. This view provides a familiar view of the Application Object Tree (AOT), which is similar to what you see in MorphX. The new AOT categorizes model element types a little differently than Microsoft Dynamics AX 2012. For example:
   - Items that were previously found in the Data Dictionary node are now under Data Model or Data Types.
   - Classes and macros are under Code.
   - Forms, menus, and other GUI elements are under User Interface.
   - Business intelligence components are under Analytics.
2. In **Application Explorer**, right-click **AOT**, and then click **Model view**.

Model view organizes programmable objects according to their model. For these tutorials, the application suite models have been removed. The core foundation and platform components have been separated from the application suite. This separation is what allowed the application suite models to be removed.

3. Double-click **Fleet Management**, or click the arrow to expand the model’s tree node. Model view provides a familiar way to work with a set of programmable objects. It’s similar to what you saw in **Classic view**, but the tree displays only the objects that are part of that particular model.
4. Double-click User Interface, and then double-click Forms to expand the forms node. In the Fleet Management sample, the names of forms and other programmable objects are prefixed with "FM" so that they're easy to identify. The name of each object is followed by the name of its layer, which in this case is "isv," and then by the name of the model it belongs to, which is "Fleet Management."

5. You can continue to expand the nodes in the tree. For example, double-click FM Rental to view the parts, data sources, and methods for the form.

6. From Application Explorer, you can open the Visual Studio code editor and view the source code for the form. For example, right-click FMRental, and then click View code.

7. You can also open the Form Designer. For example, right-click FMRental, and then click Open designer. You can expand the tree nodes in the Form Designer to see and edit the form metadata.

8. You can also preview the form in the Preview pane. Click on a control in the preview to it in the Form Designer. Similarly, click on a control in the Form Designer to highlight it in the form preview.

View the FleetManagement solution and its projects

This section of the tutorial describes the Fleet management projects and solution. Projects enable you to build your model elements (compile, synchronize the database, generate RDL files, etc.), test your application, and debug your code. We recommend that you don't make a change to a model element unless it's a part of a project; otherwise, your changes may not be compiled until you do a full build of your models, which can be a lengthy operation. In Solution Explorer, you can see the sample projects, two of which are named FleetManagement Discounts and FleetManagement Migrated. These projects are contained in a single Visual Studio solution, named FleetManagement.

1. In Solution Explorer, right-click Fleet Management Migrated, and then click Properties.
2. In the Property Pages dialog box, review the listed properties. In the Startup Object field, you can see the name of the first form that runs when you run or debug your project. You can see that the Startup Object type field is set to Form. You can also view the model name and its layer. A project always belongs to one model.
3. Don't change anything in this dialog box right now. Click Cancel to close it.
View the source file for a model element

The code in a solution is stored as XML. The following instructions show you how to view the code in Visual Studio and the source XML in Internet Explorer.

1. In Solution Explorer, be sure that the Fleet Management Migrated project node is expanded.

2. Double-click Code, and then double-click Classes, to open the folder that contains the list of classes for the Fleet Management Migrated project.

3. In the list of classes, double-click FMDDataHelper to open the code editor. Here, you can see the implementation of the FMDDataHelper X++ class.

4. Scroll down in the code to locate the main method. Tip: You can also go to the main method using the method navigation menu located on the top right of the code editor window.

If this class is set as the startup object of the project, the main method will be the execution entry point when you run or debug the project.
5. In Windows, open File Explorer, and then browse to the following folder:
C:\Packages\FleetManagement\FleetManagement\AxClass

6. Double-click the file named FMDataHelper.xml. If you’re prompted to choose a program to open the file with, click More options, and then click Internet Explorer. Otherwise, open it in Notepad.

In this file, you can see XML code that contains the metadata that describes the FMDataHelper class. For example, you can see that the class named FMDataHelper contains a set of methods. You can see the code that implements the initializeNumberSequence method for example, which is contained by an XML element. The <![CDATA]> tag ensures that the contained text isn’t interpreted or changed by the XML parser. This metadata contains the source code that you viewed in the Visual Studio code window.

When you develop a solution, you always work with code that’s stored as XML. This means that the code files are stored on your computer, not in the database. There isn’t an active connection to an application.
Build the FleetManagement migrated project

1. In the Solution Explorer, right-click Fleet Management Migrated, and then click Rebuild.

2. In the Output window, in the Show output from list, click Build. Verify that the build completed without compilation errors. Wait for the build to complete. The final build message in the Output window says, "... build completed." the final build message in the status bar (at the bottom left corner of Visual Studio) says "Ready."
3. On the **View** menu, click **Error List** to see the list of best-practice warnings. We've deliberately left some warnings in the build to demonstrate this feature.

4. Double-click any warning message to view the code or resource that caused the warning.

5. In the **Window** menu, click **Close All Documents** to close all open documents.
The application checker tool is a set of technologies that allow you to gain insight into your application code (both source and metadata) in ways that have not been possible before. The technology is based on representing both source code and metadata in XML and providing rich search facilities by using the XQuery language to express declarative queries over the source code. The current implementation runs inside a BaseX repository, which runs locally on the developer’s box.

For information about how to install and use the application checker, see Dynamics365FO-AppChecker.
This topic reviews Application Explorer, and the various views and filtering methods in it. The topic also describes how to work with elements in Application Explorer.

**Application Explorer**

Application Explorer is the tool that you use to find the elements that you want to add to a project so that you can work with them. To access Application Explorer, on the **View** menu, click **Application Explorer**. An important difference between Application Explorer and the Application Object Tree (AOT) in the MorphX environment of Microsoft Dynamics AX 2012 is that you don’t use Application Explorer to add or edit model elements. Instead, you use it to view elements, view code, find references to a selected element, and add elements to a project. To create, design, edit, and build model elements, you must use a Finance and Operations project.

**Application Explorer views**

The content in Application Explorer can be organized in two ways. In the *classic view*, all the elements from every model are grouped according to type. This view resembles the way that the AOT was organized in Dynamics AX 2012. The following illustration shows the classic view.

![Classic View](image)

The second view is called the *model view*. In this view, each model is listed separately. The elements within each model are grouped, as in the classic view. The following illustration shows the model view. Notice that the node for the Fleet Management model is expanded, and that the elements in the model are arranged as they would be in the classic view.
To switch to the model view, right-click the **AOT** node, and then click **Model view**. To switch back to the classic view, right-click the **AOT** node, and then click **Classic view**.

**Working with elements**

To work with elements, you must select them in the Application Explorer:

- To select one element, click it.
- To select a contiguous group of elements, hold down the Shift key while you click through the group of elements.
- To select noncontiguous elements, hold down the Ctrl key while you click the individual elements.

After you select the elements, right-click the selection to view the actions that you can perform. The following illustration shows the actions that are available for a form element.
The actions that are available depend on the elements that you've selected. The following are some of the common actions that you can perform for elements in Application Explorer.

**Note:** When the selected node is an element that exists in more than one model (in the case of overlayering customizations), the selected element will depend on the view that Application Explorer is currently in:

- If Application Explorer is in **model view**, the selected element is the element that belongs to the Application Explorer model that it appears under.
- If Application Explorer is in **classic view**, the selected element depends on the current context and the selected project in Solution Explorer. For example, if **Add to project** is the selected action, the selected element will be the element that belongs to the model of the current project, when applicable.

<table>
<thead>
<tr>
<th>ACTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add to project</td>
<td>Add the selected element or elements to the current project.</td>
</tr>
<tr>
<td>Add to new project</td>
<td>Add the selected element or elements to a new project.</td>
</tr>
<tr>
<td>Duplicate in project</td>
<td>Create a copy of the selected element or elements in the current project.</td>
</tr>
<tr>
<td>Customize</td>
<td>Create a customized version of the element. Click <strong>Customize</strong> when you want to do overlayering (customization) of an existing element. The model that you’re using for your current project must be in the same package as the selected element, and it must belong to a higher layer than the element that you want to customize. When you click <strong>Customize</strong>, a new “customization” model element file is created and is added to your project.</td>
</tr>
<tr>
<td>Create extension</td>
<td>Create an extension for the element. A new extension model element (.Extension) is added to the current project in Solution Explorer. This is the preferred way to work with existing elements.</td>
</tr>
<tr>
<td>ACTION</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Create extension in new project</td>
<td>Create an extension for the element as part of a new project. You define the new project when the New Project dialog box opens.</td>
</tr>
<tr>
<td>Find References</td>
<td>Find all of the X++ code and other elements that reference the selected element.</td>
</tr>
<tr>
<td>View references</td>
<td>Create a diagram that shows the other elements that reference the selected element.</td>
</tr>
<tr>
<td>Open designer</td>
<td>Open the element designer for the element of that type, so that you can view the element. Although you can modify the element, this is typically done when the element is part of a project.</td>
</tr>
<tr>
<td>View code</td>
<td>Open the code editor, where you can view and edit the X++ code for the element.</td>
</tr>
<tr>
<td>Refresh</td>
<td>Update the metadata for the element.</td>
</tr>
<tr>
<td>Compare</td>
<td>Compare the element with the XML representation of the element that you select. Typically, you will compare an element with a different version of the element from a source code control repository.</td>
</tr>
<tr>
<td>Properties</td>
<td>Open the Properties dialog box in Visual Studio, so that you can see the property settings for the element.</td>
</tr>
</tbody>
</table>

Some elements have unique commands that let you perform actions for that type of element. For example, table elements have two commands that provide additional information about the table. The first is the **View hierarchy** command. When you right-click a table element and click this command, you will see a graphical representation of the table hierarchy that the table is part of. For example, the following illustration shows the table hierarchy for the FMVehicle table.

![Table Hierarchy](image)

A similar hierarchy tool is available for classes. The second command is **Open table browser**. When you click this command, the data from the table is displayed as a list in the program.

**Filtering Application Explorer**
Application Explorer can contain a very large number of elements. This can make it difficult to find the specific element or elements that you want to work with. However, Application Explorer can filter the elements, based on a query that you supply. To filter the elements that are displayed, you can enter a query in the search bar at the top of Application Explorer.

To apply a simple filter, just type the text that you want to filter by, and then click the Filter button at the end of the search bar. For example, if you want to find all of the elements that have a name that contains the word "rental," enter rental as the search term.

To clear the filter and return to the complete view, click the Clear filter button (X) in the search bar. Notice the drop-down arrow at the end of the search bar. If you click this arrow, you will see a list of filter options that you can use to refine the filter:

- Filter By Type
- Filter By Model
- Filter By Name
- Filter By Modified Date
- Filter By Extension Point

When you select one of these options, a predefined criterion is added to the search bar. You supply the specific values for the criteria. This feature can provide powerful search capabilities. For example, if you want to find the elements that were modified within a specific period, select Filter By Modified Date, and specify the start and end dates.
Previously used filters are listed in the drop-down list at the end of the search bar. You can also perform actions on the filtered results that are displayed in Application Explorer. Right-click the AOT node, and then select one of the following actions to perform on the results.

<table>
<thead>
<tr>
<th>ACTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add search results to project</td>
<td>Add the elements from the filter results to the current project.</td>
</tr>
<tr>
<td>Add search results to new project</td>
<td>Add the elements from the filter results to a new project.</td>
</tr>
</tbody>
</table>

These actions can be used only when the filter results are limited to a single model. Your query must contain the **model**: "Model Name" criterion to limit the filter to a specific model.

**Additional resources**

- [Develop and customize home page](#)
- [Development tools in Visual Studio](#)
In this tutorial, you’ll learn about using the tools in Visual Studio to analyze and debug code in the Fleet Management application. You’ll go through a simple developer scenario in which you will set breakpoints, modify some code, and build the result.

Prerequisites

Previous experience with code and Visual Studio is helpful to get the full benefit of this tutorial. This tutorial requires you to access the environment using Remote Desktop and that you be provisioned as an administrator on the instance.

Key concepts

- The debugger in Visual Studio is used to analyze and debug code for your projects.
- The standard features of the Visual Studio debugger are available to use when you’re examining the running application. These features include modifying values of variables, setting breakpoints, and so on.
- IntelliSense and other features of Visual Studio are vital to efficient code editing and comprehension.
- Development is an iterative process. After making modifications to the code, the project will be built, and changes can be tested.

Scenario

The rental company has had unfortunate events when customers rent cars using credit cards that are past the expiration dates. You, the developer, are tasked with revising the application to help prevent this situation. You’ll identify the problem by using the debugger in Visual Studio. After the problem is identified, you’ll edit some code to implement a fix. Finally, you’ll build the project and validate that the fix was successful.

Run to a breakpoint

1. On the desktop, double-click the Visual Studio shortcut to open the development environment.

2. Open the FleetManagement solution. On the File menu, point to Open, and then click Project/Solution.

3. Browse to the desktop, and then open the FleetManagement folder. If the solution file is not on your computer, the steps to create it are listed in Tutorial: Create a Fleet Management solution file out of the Fleet Management models in the AOT.

4. Select the file named FleetManagement. The file type listed is SLN File.

5. Click Open. Loading the solution may take some time.

6. Make the FleetManagement project the startup project. In Solution Explorer, right-click the Fleet Management project, and choose Set as StartUp Project in the context menu.

7. In Solution Explorer, double-click the Fleet Management project to display its content.

8. Double-click the Code folder, and then double click the Classes folder of the Fleet Management project. Locate the FMRentailCheckoutProcessor class. Right-click this class, and then click Open. Alternatively, you can use the solution explorer search bar at the top of the solution explorer window. As you enter the name in the search bar, you’ll see the corresponding artifacts selected in the solution.
You can now see the X++ code for the class. This class has a method named `FinalizeRentalCheckout`.

9. Place a breakpoint in this method on the line following the first comment. To do this, click in the margin to the left of the line of code where you want the debugger to pause execution. You can also click anywhere in the line of code, and then press F9. The following illustration shows a breakpoint, which is displayed as a red-filled circle in the margin.

```
public boolean FinalizeRentalCheckout(FMRental frmRental) {
    Struct RentalConfirmation = new Struct("boolean OkToRent");
    // setter
    RentalConfirmation.value('OkToRent', true);

    //fire the event
    this.RentalTransactionAboutToBeFinalizedEvent(fmrentalRecord, RentalConfirmation);

    //return what survived after event handlers finishing
    return (RentalConfirmation.value('OkToRent')));
}
```

The `FinalizeRentalCheckout` method is called when a rental transaction is saved. This method calls the delegate named `RentalTransactionAboutToBeFinalizedEvent`. You can implement an event handler method, which is called by this delegate. The method that calls the delegate passes a parameter, named `RentalConfirmation`, which contains a value that indicates whether the rental should be allowed or blocked. If the rental is allowed, the value contains “true”; if it’s blocked, the value contains “false”. An event handler can change this value, based on any test the developer chooses to implement in code. In this case, we’ll modify the code to test the expiration date of the credit card.

10. Press F5 to start the application for debugging, or on the Debug menu, click Start Debugging. It’s important that you start the application in one of these ways. If you don’t, the Visual Studio debugger won’t start, so you won’t hit any of the breakpoints you’ve set. **Note:** The debugger needs to relate code position to source positions. It does this through consuming PDB files produced alongside the assemblies and net modules. The debugger will load symbols from the PDB files as described in the settings in the global tools settings. To open the options page containing the setting that controls which symbols load, go to the Tools menu and choose Options. In the Microsoft Dynamics 365 for Finance and Operations group, select the Debugging page. If this option is selected, the system will load symbols from only the PDB files related to the artifacts in the current solution. This reduces the startup time significantly, so be sure it’s selected for this lab. Be aware that when this option is selected, it won’t be possible to see source code from entities outside of the current solution. After a few moments, the browser will start and display the startup object that was selected in the project.

11. The Current Rentals page will open.
   a. In the Action Pane, click Edit.
   b. When the page is displayed, click Show list in the Show/Hide list (or press Ctrl+F8).
12. Make a change to any existing rental. For example, click Edit, and change the time that the rental period started.
13. Click Save to force a validation of the rental record. The method in which you placed a breakpoint is called. Execution pauses at the line of code that contains the breakpoint.
While the application is paused at a breakpoint, you can examine the application state. Use the same techniques that you typically would for any application developed with Visual Studio. For example, place the cursor over a variable or a parameter to see its value in a tooltip.

```csharp
public boolean finalizeRentalCheckout(PMRental fmrentalrecord)
{
    Struct RentalConfirmation = new Struct('boolean OktoRent');
    // setter
    RentalConfirmation.value('OktoRent', true);

    //fire the event
    this.RentalTransactionAboutToBeFinalizedEvent(fmrentalrecord, RentalConfirmation);

    //return what survived after eventhandlers finishing
    return ( RentalConfirmation.value('OktoRent') );
}

delegate void RentalTransactionAboutToBeFinalizedEvent(PMRental fmrentalrecord, Struct RentalConfirmation)
{
}
```

14. The other debugging tools in Visual Studio are available as well. For example, the **Locals** window shows all of the local variables for the location where execution has stopped. Click the **Locals** tab at the bottom of Visual Studio, and expand the **fmrentalrecord** variable. You will see the internal state of the record, showing the values of all the fields in the record.
Notice the value of the **Vehicle** property of the `fmrentalrecord` variable. This property is a foreign key field in the **FMRental** table. The debugger allows us to peek into the related record in the **FMVehicle** table. It shows values that belong to the **AutoIdentification** field group.

15. The **Breakpoints** window lists all of the breakpoints that have been set. Click the **Breakpoints** tab to see its content.

16. Press F10 a few times to step through the code, line-by-line, and use the full complement of debugger features. Notice that the **Locals** window updates the values of variables immediately with each statement that’s executed.

17. On the toolbar, click **Continue**, or press F5

18. Close Internet Explorer to close the **Fleet Management** application. Visual Studio will exit the debugging mode. An alternative is to choose **Stop Debugging** from the **Debug** menu. This will leave Internet Explorer open, allowing the next debugging session to start faster.

### Add the validation code

In the **FinalizeRentalCheckout** method, you saw that the developer added code to call the delegate that’s used to determine the validity of the rental. To solve the problem of expired credit cards, you’ll add an event handler,
which you’ll use to verify that the credit card isn’t expired. To simplify the lab, the handler will be added in the same file that contains the delegate. Use the following code as inspiration. Rather than copying and pasting the code, type it in manually to see the IntelliSense features in action. These features add to the high level of productivity that Visual Studio users expect.

```csharp
[SubscribesTo(classstr(FMRentalCheckoutProcessor),
delegatestr(FMRentalCheckoutProcessor, RentalTransactionAboutToBeFinalizedEvent))]
public static void RentalFinalizedEventHandler(FMRental rentalRecord, Struct rentalConfirmation)
{
    FMPaymentInformation paymentInfo;
    date ccExpiryDate, lastDayOfExpiryMonth;
    str s;

    select firstonly * from paymentInfo where paymentInfo.RecId == rentalRecord.PaymentInformationId;

    if (paymentInfo)
    {
        // Check if the payment info is valid
        // For now, we will check if the credit card is expired
        // Credit cards expire on the last day of the month indicated
        ccExpiryDate = mkdate(1, str2int(paymentInfo.ExpirationMonth), paymentInfo.ExpirationYear);
        lastDayOfExpiryMonth = endmth(ccExpiryDate);

        if (lastDayOfExpiryMonth < today())
        {
            rentalConfirmation.value('OkToRent', false);
            s = "Credit card validation failed for rental ";
        }
        else
        {
            s = "Credit card validation succeeded for rental ";
        }

        info (s + rentalRecord.RentalId);
    }
    else
    {
        rentalConfirmation.value('OkToRent', false);
        info ("No Credit card available for " + rentalRecord.RentalId);
    }
}
```

The preceding code is straightforward. The method is marked as handler for the relevant delegate by using the `SubscribesTo` attribute, as shown. In the code, the customer record is retrieved, and then the credit card date is compared to today’s date. If the expiration date of the credit card is in the past, the event handler sets a value in the `RentalConfirmation` structure to signal that the customer isn’t eligible to rent a vehicle. The idea is that any number of handlers can subscribe to the delegate. If any handler determines that a rental should not proceed, it sets the `OkToRent` flag to false. A superior implementation might refrain from doing any analysis if it determines that the `OkToRent` flag has already been set to false.

1. Be sure that you’re working in the `FMRentalCheckoutProcessor.xpp` file. Begin by adding the new event handler definition to the `FMRentalCheckoutProcessor` class. Add the following code on an empty line just above the brace (}) that marks the end of the class definition.

```csharp
public static void RentalFinalizedEventHandler(FMRental rentalrecord, Struct rentalConfirmation)
{
}
```

2. Add the attributes to the beginning of the event handler. These attributes indicate which delegate the event handler is subscribing to.
3. Now, add the code that checks the credit card expiration value. The completed method should look similar to the following code.

```
[SubscribesTo(classstr(FMRentalCheckoutProcessor),
delegatestr(FMRentalCheckoutProcessor, RentalTransactionAboutToBeFinalizedEvent))]
public static void RentalFinalizedEventHandler(FMRental rentalrecord, Struct RentalConfirmation)
{
    FMPaymentInformation paymentInfo;
    date ccExpiryDate, lastDayOfExpiryMonth;
    str s;

    select firstonly * from PaymentInfo where paymentinfo.RecId == rentalRecord.PaymentInformationId;

    if (paymentInfo)
    {
        // Check if the payment info is valid
        // For now we will check if the credit card is expired
        // Credit cards expire on the last day of the month indicated
        ccExpiryDate = mkdate(1, str2int(paymentInfo.ExpirationMonth), paymentInfo.ExpirationYear);
        lastDayOfExpiryMonth = endmth(ccExpiryDate);

        if (lastDayOfExpiryMonth < today())
        {
            RentalConfirmation.value('OktoRent', false);
            s = "Credit card validation failed for rental ";
        }
        else
        {
            s = "Credit card validation succeeded for rental ";
        }

        info (s + rentalrecord.RentalId);
    }
    else
    {
        rentalConfirmation.value('OktoRent', false);
        info ("No Credit card available for " + rentalrecord.RentalId);
    }
}
```

4. Make sure the handler and the delegate are separated by exactly one blank line.

5. On the toolbar in Visual Studio, click Save.

6. After you're satisfied with the code, build the code for the Fleet Management project. To do this, in Solution Explorer, right-click the FleetManagement project name, and then click Build. You may see errors or warnings if the code isn't correct. If so, correct the code, and build again until all warnings and errors have been resolved. You're now ready to validate that the revision works as intended.

7. On the Debug menu, click Delete All Breakpoints.

8. Place a new breakpoint in the event-handler method at the line that contains the following statement:
if (lastDayOfExpiryMonth < today())

9. Start the Fleet Management sample with debugging active by pressing F5.

10. Browse to the **Current rentals** page, as described starting in step 11 of the previous section. Select one of the reservations, and click **Edit**.

11. In the **Customer** drop-down list, select Adrian Lannin from the list, and then click **Save**. Execution pauses at the breakpoint that you set in the event-handler method.

12. Press F10 three times to step through the code block.

13. Press F5 to continue. You'll see that the customer has been disallowed.

14. In the same rental, change the customer name to Phil Spencer, and then click **Update**. This time, the transaction is allowed.

15. Close Internet Explorer.

16. Comment out the **SubscribesTo** attribute on the **RentalFinalizedEventHandler** method. This step ensures that the credit card test will no longer run as you work on the remaining tutorials.

**Best practices**

Earlier in this tutorial, you had the opportunity to add code to the project and build the solution with your changes. The build process may not have been successful, requiring you to refer to the error window. Using this window, you can get to the error by clicking on the line that describes the error. You may have noticed some diagnostic messages that don't represent compilation errors. These are diagnostics from the **Best Practice** checker. This tool will check for instances where the developer has violated a known best practice, and display a warning when one is found. The Best Practice rules apply to both code constructs and metadata. Each software development organization is likely to have its own set of best practices that they enforce. They may want to disregard some of the existing best practice checks. To support this, the set of reported best practice diagnosti
can be modified by the individual developer. To demonstrate this, complete the following steps:

1. On the View menu, click Error List. You should see a small number of best practice warnings.
3. In the Model drop-down list, make sure that the Fleet Management model is selected. The best practice rules apply to a particular model.
4. Mark the selections for some of the sets of Best Practice rules. For example, if you select the entry for CodeStyleRules, best practice guidelines for variables will be examined. After you've updated the selections, click OK.
5. Rebuild the Fleet Management project by right-clicking the project name and then clicking Rebuild. You'll notice that the violations of the best practice rules that you specified appear in the Error list window.
This topic reviews the process to build projects and full build of model packages.

The elements of a model must be built so that they can be used by the application. You can build the elements in a project. You can also build all the elements in a model. The following actions are performed during a build operation:

- Metadata validation
- X++ code validation
- Best practice checks
- Report RDL generation
- Compilation, IL generation, and creation of the .NET assemblies
- Label assembly generation and deployment of other resource files
- Database synchronization

Build a project

When you build a project, only those elements that are new or that have changed are built. To build a project, follow these steps.

1. In Solution Explorer, select the project.
2. On the Build menu, click Build <project name> to start the build process. Alternatively, right-click the project in Solution Explorer, and then click Build.

During the build process, you might notice that some elements that are built aren't part of the project. This behavior is required because of the way that assemblies are created. When you build an element, you're actually building the .NET module that the element is included in. A single .NET module contains multiple model elements, and a single assembly contains multiple .NET modules. The assembly can be created only if all the .NET modules in the assembly have been built and are up to date. If any elements in any of the .NET modules for an assembly haven’t been built or aren’t up to date, they will be built, even if they aren’t included in the current project.

**NOTE**

If you delete an element from a project, you must rebuild the project or perform a full build on the model before the deletion takes effect.

Rebuild a project

If you want to build all the elements in a project, regardless of whether they have changed, you must perform a rebuild operation. To rebuild a project, follow these steps.

1. In Solution Explorer, select the project.
2. On the Build menu, click Rebuild <project name> to start the rebuild process. Alternatively, right-click the project in Solution Explorer, and then click Rebuild.

Synchronizing the database at each build
A project property lets you specify that the synchronize operation for the database should be performed every time that you build the project. This can be useful when you’re making changes to the table structure for an application. Each time that you build, you will know that the database is synchronized with the tables as they are defined in the project. For information about how to set project properties, see Finance and Operations project type in Visual Studio. If your application has a large number of tables, and you aren’t yet testing the application, you can set the Synchronize database on build property to false. This change will reduce the time that is required to build the project. Then, when you begin testing, be sure to set this property back to true. If you must manually synchronize the tables in a project, you can right-click the project in Solution Explorer and then click Synchronize <project name> with database. To synchronize the entire database, which can be a long process, on the Dynamics 365 menu, click Synchronize database.

NOTE

If you try to synchronize the database before you have fully compiled assemblies, the Visual Studio database synchronization tool will display a message that synchronization has completed successfully, when in fact, the synchronization was not successful.

Tables and views cannot be synchronized against the database until they are fully compiled. After you complete a full build of the Application Platform, Application Foundation, and Application Suite, you can complete a Database Synchronization from the Dynamics 365 menu in Visual Studio.

Build a model’s package

You might want to build all the elements in a specific model. To do this, you must perform a full build on the package that the model belongs to. Follow these steps.

1. On the Dynamics 365 menu, click Build models.
2. In the Packages list, select the package(s) to build.
   - Package names are listed alphabetically.
   - Models belonging to the package are shown in brackets.
3. If you want to build the dependent packages first, select Build referenced packages. Any dependent package that must be built will be listed.
4. On the **Options** tab, review the options for the build process. The following options are available.

<table>
<thead>
<tr>
<th>OPTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build Pre-Compiled Forms</td>
<td>Static HTML is generated for each form during the build process. This allows faster rendering of forms at run time.</td>
</tr>
<tr>
<td>Build Reports</td>
<td>Reports are built.</td>
</tr>
<tr>
<td>Build Aggregate Measurements</td>
<td>Aggregate measurements are build.</td>
</tr>
<tr>
<td>Run Best Practice Checks</td>
<td>Best practice checks are performed during the build process.</td>
</tr>
<tr>
<td>Synchronize Database</td>
<td>The schema of the SQL database is updated during the build process (after compilation of the metadata and source code), so that it matches the metadata.</td>
</tr>
<tr>
<td>Build cross reference data</td>
<td>The data for the cross-reference feature is updated during the build process. Cross reference data enables developers to find references to code and metadata during development.</td>
</tr>
</tbody>
</table>

5. Click **Build** to start the build process.

6. Expand the **Details** tab to follow details of the build process.

**Build results**

After a build operation is completed, you will see the results in Microsoft Visual Studio. The **Output** pane in Visual Studio shows the status of the build. You can use the **Show output from** field to switch between the standard build information and the build details.
The Error List pane in Visual Studio shows the build errors and warning that occurred during the build process. If you see any build errors, you must fix them and then build again, so that valid assemblies can be created for the application. Many of the warnings that appear in the Error List pane are best practice checks that inform you of revisions that you should make to your application so that it conforms to the best practices for application development. Ideally, you should address all the best practice warnings for an application.

You can double-click most errors and warnings to see the source of the issue. The element designer or code editor will open, where you can see what property setting or code is causing the error or warning. The Task List pane in Visual Studio shows tasks that have been flagged with "TODO" comments in code. For example, the following comment indicates that some object references still require validation.

```
// TODO: validate object references
```

When the code is built, these "TODO" comments appear in the Task List pane. To view the Task List pane, on the View menu, click Task List.

To make resolution easier, you can add the elements that are affected by the error or task to the current project.
or to a new project. In the Error List pane or the Task List pane, select the rows for the errors or tasks that you want to fix, right-click, and then click **Add to project** or **Add to new project**. This saves you the effort of finding the affected elements in the application.

Additional resources

- Development tools in Visual Studio
- Develop and customize home page
This topic describes the code editor for Visual Studio.

**Code editor**

You use the code editor in Microsoft Visual Studio to write the X++ code for your applications. The X++ language is fully integrated into the Visual Studio environment. As you write your X++ code, you will see the familiar features of the Visual Studio code editor. For example, IntelliSense is displayed to help you write the code. You can also navigate to methods and classes in the code editor by using the navigation drop-down menus at the top of the code editor window.

Other features, such as collapsible sections, are also available.

**Opening the element designer**

You can open the element designer that corresponds to the current X++ source code by right-clicking in the code editor and then selecting **Open Designer**.
Additional resources

Develop and customize home page

Development tools in Visual Studio
In this tutorial, you’ll use Visual Studio’s Dynamics 365 menu to create a new model named Fleet Management tutorial. You’ll also create and edit new model elements.

Prerequisites

This tutorial requires that you have access to an environment, and that you be provisioned as an administrator.

Keywords

- **Model** - You configure your model to refer to two other models. This enables your model to reference metadata and code elements that are in other packages.
- **Project** - You create a project and then associate your project to your new model. You add elements to your project, which are also added to your model. Specifically, you add an extended data type (EDT). You also add a table that you populate with fields and a method.
- **Designer** - Each time you add an item to your project, a designer is displayed that is tailored to the item type you selected. The Properties window adjusts each time a different node of the designer is highlighted. You make updates in the designers and in the Properties window.
- **EDT** - Extended data type.

Create the Fleet Management tutorial model

1. Start Visual Studio using Run as administrator.

2. From the Dynamics 365 menu, select Model Management > Create model to open the Create model wizard.

3. Enter the following values for model parameters.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model name</td>
<td>FleetMgmtTutorial</td>
</tr>
<tr>
<td>Model publisher</td>
<td>Microsoft Corp</td>
</tr>
<tr>
<td>Layer</td>
<td>isv</td>
</tr>
<tr>
<td>Model description</td>
<td>This tutorial shows how to build the Fleet Management application by using the Microsoft Dynamics AX development tools.</td>
</tr>
<tr>
<td>Model display name</td>
<td>Fleet Management Tutorial</td>
</tr>
</tbody>
</table>
NOTE
Your model name must be FleetMgmtTutorial. Don't use any other name. In other tutorials, you'll overwrite model elements in this model by importing a project. If the model you create in this tutorial isn't named FleetMgmtTutorial, you may not be able to correctly import the project in other tutorials.

4. Click Next to advance to the next page, and then select Create New Package. The model you're creating will have its own package and build its own .NET assembly.

5. Click Next to advance to the Select referenced models step.

6. Select Application Platform and Application Foundation as referenced models.
7. Click Next to advance to the Summary step.

8. Verify the information on the summary page, and then select the Create new project and Make this my default model for new projects check boxes.


11. Select the Unified Operations template.

12. Enter the following values in the fields in the dialog box.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>FMTDataModel</td>
</tr>
<tr>
<td>Location</td>
<td>C:\FMLab</td>
</tr>
<tr>
<td>Solution</td>
<td>Add to solution</td>
</tr>
</tbody>
</table>
Create the FMTAddress extended data type

1. In Solution Explorer, right-click FMTDataModel, point to Add, and then click New Item.
2. Under Dynamics 365 Items, select Data Types.
3. Click EDT String to select the new item type.
4. In the Name field, enter FMTAddress, and then click Add.

This adds a new EDT model element to the project, and opens the EDT designer for the new element, as shown in the following illustration.
5. Select the root node of **FMTAddress** in the designer.

6. In the **Properties** window, in the **Appearance section**, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help Text</td>
<td>Check online help.</td>
</tr>
<tr>
<td>Label</td>
<td>Address</td>
</tr>
<tr>
<td>String Size</td>
<td>75</td>
</tr>
</tbody>
</table>

7. Press **Ctrl+S** to save the EDT.

**Add existing model**

Add the other required model element files to the current model and project. You can do this quickly by using the **Add existing item** feature.

1. In the **Solution Explorer**, right-click **FMTDataModel**, point to **Add**, and then click **Existing Item**.

2. Browse to **C:\FMLab\EDT\**.
Create the FMTCustomer table

1. In Solution Explorer, right-click FMTDataModel, and then click Add > New Item.

2. In the left pane, expand Installed, expand Dynamics 365 Items and then click Data Model.

3. In the list of artifacts, select Table.

4. In the Name field, enter FMTCustomer, and then click Add. The table designer opens.

Add fields to the FMTCustomer table

In the table designer for FMTCustomer, you now add several fields to the table.

3. Press Ctrl+A to select all of the files, and then click Add.
1. To add each field, right-click **Fields**, click **New**, and then select a type. As you add each field, you must specify the field name and certain other values in the **Properties** window, as described in the following table.

<table>
<thead>
<tr>
<th><strong>TYPE</strong></th>
<th><strong>FIELD NAME</strong></th>
<th><strong>PROPERTY VALUES</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>CCExpiryDate</td>
<td>Extended Data Type = FMTCCExpiryDate</td>
</tr>
<tr>
<td>String</td>
<td>Address</td>
<td>Extended Data Type = FMTAddressHelp Text = Help text for the address field.</td>
</tr>
<tr>
<td>String</td>
<td>CellPhone</td>
<td>Extended Data Type = Phone</td>
</tr>
<tr>
<td>String</td>
<td>CreditCardNum</td>
<td>Extended Data Type = FMTCreditCardNum</td>
</tr>
<tr>
<td>String</td>
<td>DriversLicense</td>
<td>Extended Data Type = FMTDriversLicense</td>
</tr>
<tr>
<td>String</td>
<td>Email</td>
<td>String Size = 80Label = Email</td>
</tr>
<tr>
<td>String</td>
<td>FirstName</td>
<td>Extended Data Type = FirstName</td>
</tr>
<tr>
<td>String</td>
<td>LastName</td>
<td>Extended Data Type = LastName</td>
</tr>
<tr>
<td>String</td>
<td>License</td>
<td>String Size = 100Label = License</td>
</tr>
<tr>
<td>String</td>
<td>Thumbnail</td>
<td>String Size = 100Label = Thumbnail</td>
</tr>
</tbody>
</table>

**TIP**

For all new fields in the table that reference an EDT, you can create the field by simply dragging the EDT element from **Solution Explorer** or **Application Explorer** and dropping it on the **Fields** node of the **FMTCustomer** table in the designer.
2. Press Ctrl+S to save the new fields on the table.

**Add fields to field groups**

1. Prepare to add some of the fields to the *AutoSummary* field group by selecting the fields in the following list. To select multiple fields, hold down the Ctrl key while you click each field:

   - Address
   - CCExpiryDate
   - CellPhone
   - CreditCardNum
   - DriversLicense
   - Email
   - FirstName
   - LastName

2. Expand the *Field Groups* node.

3. Drag the selected fields to the *AutoSummary* node

4. Use the same technique to add the fields *FirstName*, *LastName*, and *CellPhone* to the *AutoReport* field group.

5. Save the table.

**Add a method**

1. Add the X++ method named *fullName* to the *FMTCustomer* table by right-clicking the *Methods* node, and then clicking *New Method*.

2. In the code editor, replace the default method code with the following code.

```
TIP
When you type “this”, choose the field from the IntelliSense list.
```
public display FMTName fullName()
{
    return this.FirstName + ' ' + this.LastName;
}

3. Save the code.

Update the FMTAddress EDT

1. In Solution Explorer, expand the FMTDataModel project.

2. Right-click FMTAddress, and then click Open. The EDT designer opens.

3. In the EDT designer, select FMTAddress.

4. In the Properties window, in the Reference Table field, select FMTCustomer. Tip: Click the drop-down list, and then type the prefix "FMT" in the search box. This filters the drop-down list to only show tables that contain "FMT" in their name. Select the FMTCustomer table from the list of filtered entries.

5. Save the EDT.

Build the FMTDataModel project and the Fleet Management tutorial model

1. In Solution Explorer, right-click FMTDataModel, and then click Rebuild.

2. To do a full build of the entire model, on the Dynamics 365 menu, click Build models.

3. Clear the check box for all models except for Fleet Management Tutorial.

4. On the Options tab, select the Run Best practice checks check box. Note that other options available.

5. On the Models tab, click Build.

6. Click Close in the dialog box.

7. On the Window menu, click Close All Documents, to close all open documents.
This article describes the process of disabling customization of a model. By following this process, you make it ineligible for over-layering. Developers will still be able to extend that model. This article also describes how you can deprecate obsolete functionality.

Typically, you build an application in models that depend on other models. At the very least, your models will depend on the Application Platform model that is provided. Finance and Operations applications run on the Microsoft Azure cloud platform. In other words, it runs off-premises, in data centers that are managed by Microsoft. Because we can’t support changes from a large number of vendors in the fundamental models, your applications can no longer over-layer artifacts in those models. Therefore, to build your applications, you must use extensions instead of over-layering. Even though all the artifacts in the models that you depend on are available for documentation purposes, you can’t compile someone else’s intellectual property and run it in the cloud.

**Locking customizations**

Transitioning a model from over-layering to extensions involves three steps:

1. You set a property that causes instances of over-layering to be flagged as warnings. This step is sometimes known as “soft-locking.”
2. You have a period when you can burn down the warnings by using extensions instead of over-layering.
3. When you have resolved all the warnings, it’s time to make over-layering a hard error that causes compilation to fail. This step is known as “hard-locking.” When a model is hard-locked, the tooling that is required for over-layering can’t be used for that model. Additionally, you can’t have more than one model in a given package.

Currently, there is no tooling that you can use to manipulate the property that disables customizations. Instead, you must add the `Customization` XML element to the model descriptor file, as shown in the following example. You can find the model descriptor file at ...\<packages>\<package name>\Descriptor\<model name>.xml.

```xml
<?xml version="1.0" encoding="utf-8"?>
<AxModelInfo xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
 <Customization>DoNotAllow</Customization>
 <Description>My cool locked application</Description>
 ...
</AxModelInfo>
```

There are three levels of customization settings:

- **Allow customizations** – If the `Customization` element is omitted, or if it’s provided but the text `Allow` is used inside it, there are no restrictions.
- **Soft-locking** – If you require soft-locking, use the text `AllowAndWarn` inside the `Customization` element.
- **Hard-locking** – To disable customization of the model, use the text `DoNotAllow` inside the `Customization` element, as shown in the preceding example.
Backward compatibility of the platform

There is a requirement that your application must continue to run even if a newer version of a dependent platform model is installed. Therefore, we enforce strict requirements for backward compatibility on all the changes that we make in those platform models. We can clarify this point through an example. For this example, you have a base model M that has the following class.

```csharp
public void DoSomething(int arg)
{
    // Do great stuff
}
```

In your model that depends on model M, you want to take advantage of all the great functionality that the `DoSomething` class offers. Therefore, you have the following code.

```csharp
{
    var c = new SomeClass();
    c.DoSomething(42);
}
```

Later, your vendor (perhaps Microsoft) releases a new, updated version of the model that you depend on (model M). When this model is released, we must support the public interface, because things must run without recompilation. Suppose that we changed the method signature by adding another parameter, as shown here.

```csharp
public void DoSomething(int arg, str anotherArg)
{
    // Do greater stuff
}
```

In this case, we break the app. The common language runtime (CLR) can’t call the method, because the parameter profile that is assumed in the call isn’t the same as the parameter profile of the callee (the calling code provided one parameter, but two are now required). Therefore, it’s a good idea to limit what is publicly consumable. If something is private, nobody can use it from outside your model. However, another option is to make the artifact internal. If you apply the `internal` modifier to a class or method, the artifact is visible only inside your model and can’t be reached from the outside. Essentially, you must assume that anything that isn’t internal or private will be used from outside your model. Therefore, you must support it forever. Never make anything public or protected unless it absolutely must be public or protected, and use interfaces to hide implementation details that aren’t relevant outside the boundary of your model. There is no way to mark metadata (as opposed to code) with the internal visibility specifier.

Testing internal functionality

By limiting the surface area of your model, you help guarantee that you will be able to fix issues in your models that do not allow customizations, and that the application will run smoothly. However, you might argue that, by making things internal, you limit the ability to test your code. To remedy this situation, you can inform your test packages about the packages that they should test. In other words, the test packages can access internal things. To implement this solution, you edit the descriptor file, as shown in the following example.
You can provide any number of external packages by including them in the `InternalsVisibleTo` element. This information is provided for the package that contains the code to test, not for the package that contains the testing code. In other words, the package determines who it shares information with. **Note:** The elements under `AxModelInfo` must be in alphabetical order.

### Deprecating functionality

**Deprecating methods**

Sometimes, you no longer want to support source code that is public. As we mentioned earlier, you should not remove an artifact from the code unless that artifact is private or internal, because there might be users who rely on it. Instead, you can use the `SysObsoleteAttribute` attribute to specify that consumers should no longer use that artifact. We recommend that you mark things as obsolete in two phases:

1. Specify that using the artifact is flagged as a warning.
2. After one or more release cycles, make using the artifact an error.

In the following example, the `DoSomething` method is defined in the first release of a model.

```csharp
class SomeApi
{
    void DoSomething()
    {
        // Do great stuff
    }
}
```

In the second release, you’ve determined that there is a better way to accomplish something. Therefore, you add the new implementation and make the old implementation obsolete.

```csharp
class SomeApi
{
    [SysObsolete("Use DoSomethingNew instead", false)]
    void DoSomething()
    {
        // Do great stuff
    }

    void DoSomethingNew()
    {
    }
}
```

All your existing customers will continue to call the old version. This situation is fine, but the customers won’t be able to take advantage of the benefits of the new version. Anyone who codes against your class will receive a build warning together with the message that you provided in the `SysObsolete` attribute argument. Presumably, these clients will update their code so that it uses the new method. As time passes, more clients will move to the new version. Therefore, at some point, it will make sense for you to make coding against the method a hard error.
class SomeApi
{
    [SysObsolete("Use DoSomethingNew instead", true)]
    void DoSomething()
    {
        // Do great stuff
    }

    void DoSomethingNew()
    {
    }
}

Again, for the reasons that we mentioned earlier, you may never be able to get rid of the old method completely, because it was not made private or internal.

**Deprecating metadata**

For deprecating model elements (tables, data entities, EDTs, Enums, ...etc.), use the property `IsObsolete` that is available on all model element types. `IsObsolete` is also available on table, view, and data entity fields. When you set `IsObsolete` to Yes, references to that element or field will cause compilation warnings.
This topic describes how to generate a Customization Analysis Report for your model. It also describes some best practice rules that are included in the report, and provides suggestions for fixing errors and warnings that are associated with these rules.

**What is the Customization Analysis Report?**

The Customization Analysis Report is a tool that analyzes your customization and extension models, and runs a predefined set of best practice rules. The report is one of the requirements of the solution certification process. The report is in the form of a Microsoft Excel workbook.

**How to generate the report**

The xppbp.exe tool is located in c:\packages\bin or I:\AosService\PackagesLocalDirectory\bin. To generate the Customization Analysis Report, run the following command in a development environment.

```
xppbp.exe -metadata=<local packages folder> -all -model=<ModelName> -xmlLog=C:\BPCheckLogcd.xml -module=<PackageName> -car=<reportlocation> -packagesroot=K:\AosService\PackagesLocalDirectory
```

If your custom model references an ISV model, then you must include the **-PackagesRoot** parameter, for example:

```
-xppbp.exe -metadata=C:\Packages -all -model="MyAppSuiteCustomizations" -xmlLog=C:\temp\BPCheckLogcd.xml -module="ApplicationSuite" -car=c:\temp\CARreport.xlsx
```

**Issues List**

This section describes all best practice rules (errors, warnings, or informational messages) that can appear on the **Issues List** page of the report and provides suggestions for fixing the issues. Issues are of the **metadata** or **code** type. For all **code** issues, keep in mind that, if the warning or error occurs in a method that you've overlaid, the lines of code that violate the rule might belong to a model in a lower layer. In that case, you're not responsible for fixing warnings and errors in code that isn't yours.

**BPCheckPackReturnsConnull**

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>THIS RULE APPLIES TO ALL CLASSES THAT IMPLEMENT THE SYSPACKABLE INTERFACE. IT MAKES SURE THAT THE PACK METHOD DOESN'T RETURN CONNULL().</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error message</td>
<td>Container pack method returns connull in a Runbase derived class</td>
</tr>
<tr>
<td>Issue type/severity</td>
<td>Code/Warning</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>THIS RULE APPLIES TO ALL CLASSES THAT IMPLEMENT THE SYSPACKABLE INTERFACE. IT MAKES SURE THAT THE PACK METHOD DOESN’T RETURN CONNULL().</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>How to fix it</td>
<td>Update the return value of the Pack method, or return Super() when applicable.</td>
</tr>
</tbody>
</table>

**BPCheckParametersModified**

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>THIS RULE FAILS IF A PARAMETER OF A METHOD IS MODIFIED INSIDE A METHOD.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error message</td>
<td>Parameter 1% in method %1 are modified inside the method</td>
</tr>
<tr>
<td>Issue type/severity</td>
<td>Code/Warning</td>
</tr>
<tr>
<td>How to fix it</td>
<td>Refactor your code. Parameters must be modified by the caller, not within the called method.</td>
</tr>
</tbody>
</table>

**BPCheckSQLCode**

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>THIS RULE FAILS IF YOUR X++ CODE CONTAINS DIRECT SQL CODE.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error message</td>
<td>SQL code found in method %1</td>
</tr>
<tr>
<td>Issue type/severity</td>
<td>Code/Warning</td>
</tr>
<tr>
<td>How to fix it</td>
<td>Refactor your code to use X++ to access the database.</td>
</tr>
</tbody>
</table>

**BPCheckNestedLoopInCode**

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>THIS RULE FAILS IF IT FINDS A WHILE SELECT LOOP NESTED WITHIN A LOOP.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error message</td>
<td>Nested data access loop found in %1 method</td>
</tr>
<tr>
<td>Issue type/severity</td>
<td>Code/Warning</td>
</tr>
<tr>
<td>How to fix it</td>
<td>Refactor your code to use joins instead of nested data access loops. If you can't refactor the code without altering the business logic of your method, document an exception when you submit your Customization Analysis Report to Microsoft.</td>
</tr>
</tbody>
</table>

**BPCheckInsertMethodInLoop**

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>THIS RULES FAILS IF IT FINDS A CALL TO THE METHOD INSERT NESTED INSIDE A LOOP. WE RECOMMEND THAT YOU USE INSERTRECORDLIST. THIS RULE DOESN’T APPLY TO INMEMORY TABLES.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error message</td>
<td>Insert method can be replaced with RecordInsertList in method %1</td>
</tr>
<tr>
<td>Issue type/severity</td>
<td>Code/Warning</td>
</tr>
</tbody>
</table>
How to fix it

Refactor your code to use set-based operations, such as InsertRecordList.

BPCheckSelectwithJoin

DESCRIPTION

This rule fails if it finds nested select statements that aren't joined.

Error message

Nested select statement in %1 method can be joined

Issue type/severity

Code/Warning

How to fix it

Refactor your code to use joins instead of a nested select statement. If you can't refactor the code without altering the business logic of your method, document an exception when you submit your Customization Analysis Report to Microsoft.

BPFunctionCallwithSelect

DESCRIPTION

This rule fails if a function call is found within a select statement.

Error message

Function call found in select statement in method %1

Issue type/severity

Code/Warning

How to fix it

Assign the function's return value to a local variable before you call the select statement, and use the local variable in the select statement.

BPCheckInvalidExecuteQuery

DESCRIPTION

This rule fails if a call to Addrange is found after a call to super() in the ExecuteQuery method.

Error message

Add range after super() should not be added in ExecuteQuery method for form %1

Issue type/severity

Code/Warning

How to fix it

Refactor your code to avoid this pattern.

BPCheckInvalidInitFormMethod
### BPCheckEmptyLoop

**DESCRIPTION**
This rule fails if it finds loops that have no statements.

<table>
<thead>
<tr>
<th>Error message</th>
<th>Empty loop found in method %1 in class %2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue type/severity</td>
<td>Code/Warning</td>
</tr>
<tr>
<td>How to fix it</td>
<td>Remove the loop from your code.</td>
</tr>
</tbody>
</table>

### BPCheckLockQueryRange

**DESCRIPTION**
This rule fails if the code calls ADDRANGE in the INIT method of the form, and doesn't set the range as locked or hidden.

<table>
<thead>
<tr>
<th>Error message</th>
<th>Range should be locked or hidden in form %1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue type/severity</td>
<td>Code/Warning</td>
</tr>
<tr>
<td>How to fix it</td>
<td>Add QueryBuildRange.status(RangeStatus::Locked) or QueryBuildRange.status(RangeStatus::Hidden) after the call to AddRange.</td>
</tr>
</tbody>
</table>

### BPCheckSkipStatementValidation
### Description

This rule reports an informational message if it finds a set-based operation that doesn't have `Skip` statements.

- When `update_recordset` is found, the rule checks for `skipDataMethods(true)`. The rule applies only when the `update` method on the table is overridden.
- When `insert_recordset` is found, the rule checks for `skipDataMethods(true)`. The rule applies only when the `insert` method on the table is overridden.
- When `delete_from` is found, the rule checks for `skipDeleteActions(true)`. The rule applies only when the `insert` method on the table is overridden.

This is an informational message that highlights the need to call `skip` methods to take full advantage of the performance gains of set-based operations. If `skip` methods are not used, the execution falls back to a row-by-row operation.

### Error message

Set-based operation must invoke Skip statements in method %1 in class %2; otherwise, execution will fall back to a row by row operation.

### Issue type/severity

Code/Information

### How to fix it

Use `skipDataMethods(true)` or `skipDeleteActions(true)` when applicable.

### BPCheckNoTTSTryBlock

**DESCRIPTION**

THIS RULE FAILS IF IT FINDS A TRY STATEMENT WITHIN A TTS BLOCK THAT ISN'T HANDLED CORRECTLY.

### Error message

Tts block with Try statement does not explicitly catch exceptions in method %1

### Issue type/severity

Code/Warning

### How to fix it

Use the code example that follows this table.

The following examples show when the rule will fail or pass. Use these examples as guidelines to refactor your code.

```java
ttsbegin;
  try {
  } // fail
  catch { }
  try {
  } // pass
  catch(Exception::UpdateConflict) {
  }
  try {
  } // pass
  catch(Exception::UpdateConflictNotRecovered) {}
```

### BPCheckEmptyTableMethod
### BPCheckBatchJobsEnabled

**Description**
This rule makes sure that all classes that extend RUNBASEBATCH have a CANGOBATCH method that returns true.

<table>
<thead>
<tr>
<th>Error message</th>
<th>Custom job is not batch-enabled in class %1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue type/severity</td>
<td>Code/Warning</td>
</tr>
<tr>
<td>How to fix it</td>
<td>The canGoBatch method must return true for all batch classes.</td>
</tr>
</tbody>
</table>

### BPCheckDisplayMethodCached

**Description**
For each control that is bound to a data method, this rule fails if one of these conditions is met:
- The Cache Data Method property is set to Auto, the corresponding table display/edit method doesn't have the SysClientCacheDataMethodAttribute, and the init method of the data source doesn't use CacheAddMethod.
- The Cache Data Method property is set to No, and the init method of the data source doesn't use CacheAddMethod.

<table>
<thead>
<tr>
<th>Error message</th>
<th>Display method %1 in form %2 not cached</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue type/severity</td>
<td>Code/Warning</td>
</tr>
<tr>
<td>How to fix it</td>
<td>Use the note that follows this table.</td>
</tr>
</tbody>
</table>

You can fix this warning by using one of the following patterns:

- Set the Cache Data Method property to Yes.
- Set the Cache Data Method property to Auto, and mark the data method of the table with the SysClientCacheDataMethodAttribute attribute. Here is an example:

```java
[SysClientCacheDataMethodAttribute(true)]
Display TransDate myDateMethod()
{
   //...
}
```
- Use `CacheAddMethod` in the `init` method of the form to mark the method as cached.

### BPCheckSQLQueryInInit

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>THIS RULE FAILS IF A DATA ACCESS QUERY THAT HAS A WHILE LOOP IS FOUND IN THE INIT METHOD OF A FORM. THIS PATTERN COULD CAUSE PERFORMANCE ISSUES WHEN THE FORM IS INITIALIZED.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error message</td>
<td>SQL query with while loop was found in init method of form %1</td>
</tr>
<tr>
<td>Issue type/severity</td>
<td>Code/Warning</td>
</tr>
<tr>
<td>How to fix it</td>
<td>Refactor your code to avoid this pattern.</td>
</tr>
</tbody>
</table>

### BPCheckNewQueryWithForm

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>THIS RULE FAILS IF IT FINDS THE NEW QUERY() STATEMENT IN THE INIT OR EXECUTEQUERY METHOD OF A FORM.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error message</td>
<td>Data source query overridden in form method %1</td>
</tr>
<tr>
<td>Issue type/severity</td>
<td>Code/Warning</td>
</tr>
<tr>
<td>How to fix it</td>
<td>Refactor your code to avoid this pattern.</td>
</tr>
</tbody>
</table>

### BPCheckSelectForUpdateAbsent

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>THIS RULE FAILS IF SELECT FORUPDATE IS USED TO SELECT RECORDS, BUT AN UPDATE ISN'T BEING PERFORMED. THIS RULES DOESN'T APPLY TO TABLES THAT ARE ENABLED FOR OPTIMISTIC CONCURRENCY CONTROL (OCC).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error message</td>
<td>Select ForUpdate not implemented in method %1</td>
</tr>
<tr>
<td>Issue type/severity</td>
<td>Code/Warning</td>
</tr>
<tr>
<td>How to fix it</td>
<td>Use <code>Select</code> instead of <code>Select ForUpdate</code>, or set the OCC Enabled property to Yes on the table.</td>
</tr>
</tbody>
</table>

### BPCheckTablePropertyMismatch

| Description | This rule fails when the table belongs to a table group but doesn't have the appropriate Cache Lookup value. The rule fails if one of these conditions is met:  
- The Table Group property is set to Main, and the Cache Lookup property is set to NotinTTS or EntireTable.  
- The Table Group property is set to Group or Parameter, and the Cache Lookup property is set to NotinTTS.  
- The Table Group property is set to WorksheetHeader, WorksheetLine, or Transaction, and the Cache Lookup property is set to Found, FoundAndEmpty, or EntireTable. |

<table>
<thead>
<tr>
<th>Error message</th>
<th>%1 table has an invalid combination of table group and cache lookup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue type/severity</td>
<td>MetaData/Warning</td>
</tr>
<tr>
<td>How to fix it</td>
<td>Set an appropriate Cache Lookup value on the table.</td>
</tr>
</tbody>
</table>

**BPCheckMissingDeleteActions**

**DESCRIPTION**

This rule validates that the value of either a delete action or the on delete property of a table relation isn't equal to None. The rule isn't triggered when the related or current table is a Tempdb, In Memory Table, Reference Table, Staging Table, or Parameter Table.

<table>
<thead>
<tr>
<th>Error message</th>
<th>Delete actions missing in table %1 which is related to table %2 with relation name %3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue type/severity</td>
<td>MetaData/Warning</td>
</tr>
<tr>
<td>How to fix it</td>
<td>Set a delete action value that isn't equal to None.</td>
</tr>
</tbody>
</table>

**BPCheckAddressModel**

**DESCRIPTION**

This rule fails if a table field is of the addressZipcodeId or addressStateId type. These types indicate that the code didn't uptake the newer address framework.

<table>
<thead>
<tr>
<th>Error message</th>
<th>Field %1 of table %2 is not part of address location model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue type/severity</td>
<td>MetaData/Warning</td>
</tr>
<tr>
<td>How to fix it</td>
<td>Replace these types with any other appropriate EDT type in the Directory model.</td>
</tr>
</tbody>
</table>

**BPCheckDimensionModel**

**DESCRIPTION**

This rule fails if a table field is of the dimension or ledgerAccount type. These types indicate that the code didn't uptake the newer financial dimension framework.

<table>
<thead>
<tr>
<th>Error message</th>
<th>Field %1 of table %2 is not part of financial dimension framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue type/severity</td>
<td>MetaData/Warning</td>
</tr>
<tr>
<td>How to fix it</td>
<td>Replace these types with any other appropriate EDT type in the Dimensions model.</td>
</tr>
</tbody>
</table>

**BPCheckNumberOfNewFields**
<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>THIS RULE VERIFIES THAT NO MORE THAN 10 FIELDS WERE ADDED TO A TABLE DURING THE PROCESS OF CUSTOMIZING OR EXTENDING THAT TABLE. THIS RULE DOESN'T APPLY TO STAGING TABLES.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error message</td>
<td>Number of new fields in table %1 is greater than</td>
</tr>
<tr>
<td>Issue type/severity</td>
<td>Metadata/Warning</td>
</tr>
<tr>
<td>How to fix it</td>
<td>Refactor your schema and create related tables instead of adding too many fields to an existing table.</td>
</tr>
</tbody>
</table>

**BPCheckEnumUpgradeIssue**

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>THIS RULE FAILS WHEN THE FOLLOWING CONDITION IS MET: WHEN YOU CUSTOMIZE AN ENUM (OVERLAYER), NEW ENUM VALUES ARE LESS THAN THE MAXIMUM EXISTING VALUE + 10. THIS RULES DOESN'T APPLY TO EXTENSIBLE Enums.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error message</td>
<td>Enum %1 will have upgrade issues</td>
</tr>
<tr>
<td>Issue type/severity</td>
<td>MetaData/Warning</td>
</tr>
<tr>
<td>How to fix it</td>
<td>Use larger enum values, or extend the enum.</td>
</tr>
</tbody>
</table>

**BPCheckPassiveJoinUse**

| Description | This rule validates that, when a form allows for pre-loading of data, the link type of tab page data sources is passive. The rule fails if all the following conditions are met:  
• A form has the AllowPreLoading property set to Yes.  
• A tab page on the form is bound to a top-level data source.  
• The data source's Join Source property is set.  
• The data source's Link Type property isn't set to Passive. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Error message</td>
<td>New message suggest: “Use passive joins on data sources %1 bound to a tab page control in form %2”</td>
</tr>
<tr>
<td>Issue type/severity</td>
<td>MetaData/Warning</td>
</tr>
<tr>
<td>How to fix it</td>
<td>Set the AllowPreLoading property to No on the form, or use passive joins on the data source. Passive joins require that you explicitly program when the query of a tab page is run.</td>
</tr>
</tbody>
</table>

**BPCheckAlternateKeyAbsent**

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>THIS RULE VERIFIES THAT TABLES THAT HAVE A UNIQUE INDEX HAVE AT LEAST ONE ALTERNATE KEY.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error message</td>
<td>Table %1 does not have an alternate key</td>
</tr>
<tr>
<td>Issue type/severity</td>
<td>MetaData/Warning</td>
</tr>
</tbody>
</table>
### How to fix it

Set the **Alternate Key** property to **Yes** on a unique index of the table.

---

#### BPCheckBaseTableModified

**DESCRIPTION**

**THIS RULE VERIFIES THAT TABLES THAT HAVE A UNIQUE INDEX HAVE AT LEAST ONE ALTERNATE KEY.**

**Error message**

%1 is a base table and must not be modified

**Issue type/severity**

MetaData/Warning

**How to fix it**

Don't customize the table.
This topic reviews the element designers and explains how to use them. The tools contain designers for each kind of element in the program. You use these designers when you create or modify elements.

Open an element designer

To open an element designer, follow these steps.

1. Find the element in the project or in Application Explorer.
2. Right-click the element. In Application Explorer, click Open designer. In the project, click Open.
3. Expand the nodes in the element designer to see the details about the element.

Node properties

When you select the individual nodes in the element designer, the Properties pane in Visual Studio shows the various properties for that node. Most of the characteristics of an element are controlled by these properties. For example, the following illustration shows the element designer for the FMCustomer table. Notice that the top-level node is selected.

The following illustration shows the set of properties for the table, which corresponds to the top-level node that is selected.
Each node for the element will have a set of properties that applies to it. To make it easier to find the properties that you want to work with, use the buttons at the top of the **Properties** pane to control how they are displayed. The properties can be arranged in the following ways.

<table>
<thead>
<tr>
<th>ORGANIZATION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabetical</td>
<td>Arrange the properties in alphabetical order.</td>
</tr>
<tr>
<td>Categorized</td>
<td>Arrange the properties into standard categories for the node type.</td>
</tr>
<tr>
<td>Changed</td>
<td>Divide the properties into those that have been changed and those that use the default values.</td>
</tr>
<tr>
<td>Frequency</td>
<td>Divide the properties into categories, based on whether a property is often, occasionally, or rarely changed.</td>
</tr>
</tbody>
</table>

**Working with nodes**

When you create or modify elements, you will often find that you must add or remove nodes for the element. For example, to add a field to a table, right-click the **Fields** node for the table element, point to **New**, and then click the type of field to add.
To remove a node, right-click the node, and then click **Delete**. You can also perform other actions for a node. You can rename a node, duplicate a node, or move the node up or down in the node list.

### Searching element nodes

Sometimes, the node list for an element can be long, so it’s difficult to find the specific node that you’re looking for. Notice that there is a search bar at the top of each element designer. You can enter a string to search for, and the node list will be filtered to include only the nodes that match the search string. For example, the following illustration shows the element designer for the FMCustomer table after the “Email” search term was applied. Only nodes that have names that match that search term appear in the element designer.

If you’re working with a customization element or an extension element, you can prefix your search string with `c:` (for “customization elements”) or `e:` (for “extension elements”) to return only customizations or extensions, respectively. **Examples**

- **e:** returns all extensions that belong to the current element.
- **c:** returns all customizations that belong to the current element.
- **e:Email** returns all extension nodes that have the string “Email” in their name.
- **c:Email** returns all customization nodes that have the string “Email” in their name.

### Navigating to related elements

The value of a node in the element designer is often a reference to another element. For example, a field node in a table element is typically based on an extended data type (EDT) element. When you right-click a node in the element designer, you can click the **Go to <element>** command to navigate to that related element. For example, when you right-click the **FuelType** node in the list of fields for the FMVehicle table, you can click **Go to Base Enum FMFuelType** to show the base enumeration that is used to define the field.
This article reviews the commands that have been added to the Microsoft Visual Studio Tools to help you determine how elements are used in an application.

Because of the large number of elements in a typical application, commands have been added to the Microsoft Visual Studio Tools to make it easier to determine how an element is used.

**Finding where elements are used**

During build operations, cross-reference information is generated that can be used to show how elements are used. You can right-click an element and then click **Find References** to display a list of the locations where that element is used. When you click one of the items in the list, the designer for the element opens.

**Viewing a reference diagram**

When you right-click some higher-level elements, such as tables, the **View Reference** command is available. This command produces a graphic that shows the elements that are related to the current element. You can right-click the items in the graphic and then click **Go To Definition** to navigate to those elements.

**Additional resources**

*Development tools in Visual Studio*
Element designers
Model files let you distribute models to customers and partners, and can be installed in development environments. They are key components of a Lifecycle Services (LCS) solution. Model files contain a model descriptor file, metadata, source code, and referenced .NET assemblies (when applicable). This article describes how to export a model into a model file, install a model file, and delete a model in a development environment.

Export a model into a model file for distribution

To export an existing model into a model file, use the ModelUtil.exe tool and the `-export` directive. This tool is located in the packages bin folder (typically, `c:\packages\bin` or `i:\AosService\PackagesLocalDirectory\bin`).

Example

```
ModelUtil.exe -export -metadatastorepath=[path of the metadata store] -modelname=[name of the model to export] -outputpath=[path of the folder where the model file should be saved]
```

The preceding example creates an .axmodel file under `c:\temp`. Typically, you then upload the model file to the Asset Library of the customer project or the Microsoft Dynamics Lifecycle Services (LCS) solution project.

Install a model in a development environment

To install a model file in a development environment, use the ModelUtil.exe tool and the `-import` directive.

```
ModelUtil.exe -import -metadatastorepath=[path of the metadata store where model should be imported] -file= [full path of the file to import]
```

If the model already exists in your development environment, you must first delete it by using the `-delete` directive.

```
ModelUtil.exe -delete -metadatastorepath=[path of the metadata store] -modelname=[name of the model to delete]
```

NOTE

If you're using an older version, you can use the -replace parameter to replace standard models (like Foundation) for overlayering.

Resolve conflicts

If you install a model on a development environment that contains customizations to that model (in a higher-layer), you may have to resolve code or metadata conflicts. You can use the development tools to create a project that groups all items that have conflicts.

1. Under Dynamics 365 > AddIns, click Create Project from Conflicts.
2. In the dialog box, select the model to check for conflicts. This is the model that contains customizations to elements in the newly installed baseline model.

3. Click **Create project**. A project is generated that contains only the elements in that model that have conflicts.

4. Open the designer for the conflicting element to view and resolve conflicts by using the tools that are provided.
This article describes how to use metadata search to search your code and metadata for arbitrary patterns and content.

Given the large volume of the code base and metadata, it is often necessary to find things in the code that meet a certain criteria. For example, you might not know the name of the metadata element that contains the pattern or meets the criteria. Metadata search is exposed in Visual Studio through two user interfaces: the Metadata Search tool window and the Navigate To window.

**Metadata search tool window**

You can access the Metadata search tool window from the Dynamics 365 > Metadata Search menu command. Enter your search query to start the search. Results will start populating in the window asynchronously as you type. You can double-click any result line to navigate to the corresponding X++ code or metadata that matches your search query.

You can also select one or more results, right-click, and then add these elements to a project. You don’t need to wait for the search to complete before you start interacting with the search results.

**Navigate To window**

The Navigate To window is invoked using the Ctrl+, (the comma character) shortcut keys. Pressing Ctrl+, displays the query entry box in top-right corner of the Visual Studio main document window. You can also
access the **Navigate To** window from the Visual Studio **Edit** menu. Enter you search query and see the results appear as you type. A progress indicator will stop when the search is complete. You don’t need to wait for the search to complete to start interacting with the results.

---

**Search query syntax**

This section describes the search query syntax and provides example queries.

**Syntax**

The search query is a search string that consists of a set of filters in this general form:

```
<filter_1>:<filter_1_value> [<filter_2>:<filter_2_value> ... [ <filter_N>:<filter_N_value>]]
```

Where `<filter_i>` is one of the acceptable filter names, and `<filter_i_value>` are comma separated (and possibly quoted) filtering values.

**Filter names**

- **Name**: Filter by element name. This is the default filter, meaning if you just type a filter value, it is assumed to be an element name. Each comma-separated value is an acceptable element name.

- **Type**: Filter by element type. Each comma-separated value should be the name of an element or subelement type (root type or subtype) (that is, table, class, field). Logic of filtering is:

  \[(\text{roottype}_1 \text{ OR roottype}_2 \text{ OR } \ldots \text{ OR roottype}_N) \text{ AND (} \text{subtype}_1 \text{ OR subtype}_2 \text{ OR } \ldots \text{ OR subtype}_N)\]\n
- **Model**: Filter by model name. Each comma-separated value should be the name of a model in your application.

- **Property**: Apply property filters. Each comma-separated value should be in the form `property_name=property_value`.

- **Code**: Filter using code snippets, use quotes around code snippets. The matching source code is the elements that contain the specified code snippet.
You can get help about using filter and filter syntax by opening the drop-down menu available in the search box.

### Examples

<table>
<thead>
<tr>
<th>QUERY STRING</th>
<th>WHAT IT DOES</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrvExpTable</td>
<td>If the token is by itself, it is assumed to be the name. So this will find everything in the application that has &quot;TrvExpTable&quot; in the name.</td>
</tr>
<tr>
<td>type:form ccount</td>
<td>Finds all forms that have &quot;ccount&quot; in their names.</td>
</tr>
<tr>
<td>type:form property:formtemplate=listpage</td>
<td>Finds all forms that contain the property &quot;FormTemplate&quot; equal to 'ListPage'.</td>
</tr>
<tr>
<td>type:table,formDesign property:&quot;WorkflowDataSource=TrvExpTable&quot;</td>
<td>Finds formDesign nodes under tables, nothing would be found.</td>
</tr>
<tr>
<td>type:form,formmenufunctionbuttoncontrol property:Text=@SYS311998</td>
<td>Finds all menu function button controls with the Text property equal to (a label) @SYS311998'.</td>
</tr>
<tr>
<td>type:table,method name:insert</td>
<td>Finds tables with a method containing “insert” in the method name.</td>
</tr>
<tr>
<td>type:table,tableindex name:Export</td>
<td>Finds tables with an index name containing the word “Export”.</td>
</tr>
<tr>
<td>type:table,tableindexfield name:xpNum</td>
<td>Finds table indexes with “xpNum” in the index field name.</td>
</tr>
<tr>
<td>type:table,tablefieldgroup name:EPNew</td>
<td>Finds FieldGroups (in tables) containing ‘EPNew’ in their names.</td>
</tr>
<tr>
<td>type:form,formgridcontrol property:allowedit=no,heightmode=column</td>
<td>Finds form grid controls, with properties allowedit equal to &quot;no&quot; and heightmode equal to &quot;column&quot;.</td>
</tr>
<tr>
<td>type:form,formtabcontrol property:arrangeMethod=Vertical,ViewEditMode=view,WidthMode=Auto</td>
<td>Finds form tab controls, with properties arrangeMethod equal to &quot;Vertical&quot; and ViewEditMode equal to &quot;view&quot; and WidthMode equal to &quot;Auto&quot;.</td>
</tr>
<tr>
<td>type:form,formDesign property:&quot;WorkflowDataSource=TrvExpTable&quot;</td>
<td>Finds all forms with the &quot;WorkflowDataSource&quot; property in the FormDesign node set to the value &quot;TrvExpTable&quot;.</td>
</tr>
<tr>
<td>model:&quot;Application Suite&quot; type:formdesign property:style=simplelistdetail</td>
<td>Find all forms in Application Suite model that has the style property set to simpleListDetail in the FormDesign node.</td>
</tr>
<tr>
<td>code:&quot;return null&quot;</td>
<td>Finds all places in the source code that contains &quot;return null&quot;.</td>
</tr>
<tr>
<td>code:&quot;element.lock()&quot; type:form</td>
<td>Finds all places in the forms source code that contain the snippet &quot;element.lock()&quot;.</td>
</tr>
<tr>
<td>QUERY STRING</td>
<td>WHAT IT DOES</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>code:=&quot;insert&quot; type:table,form</td>
<td>Finds all places in the source code of either forms or tables that contain &quot;insert&quot;.</td>
</tr>
<tr>
<td>code:=&quot;public display&quot; type:form,method</td>
<td>Finds all form methods that contain the code &quot;public display&quot;.</td>
</tr>
<tr>
<td>type:formbuttoncontrol property:text=</td>
<td>Finds all form Button Controls that have empty text properties.</td>
</tr>
</tbody>
</table>
This topic describes the concept of models and packages. It also explains how to use the development tools in Microsoft Visual Studio to create new models, how to update the parameters of existing models, and how to visualize dependencies between models.

To work with models in the model store, you use tools in Microsoft Visual Studio. You can create new models and change parameters for existing models.

Conceptual overview

A model is a group of elements, such as metadata and source files, that typically constitute a distributable software solution and includes customizations of an existing solution. A model is a design-time concept, for example a warehouse management model or a project accounting model. A model always belongs to a package.

A package is a deployment and compilation unit of one or more models. It includes model metadata, binaries, and other associated resources. One or more packages can be packaged into a deployable package, which is the vehicle used for deployment on runtime environments.

Creating a new model

You use the Create model wizard to create new models. You can access this wizard from Model Management on the Dynamics 365 menu. You can create two types of models:

- **A model that is deployed in its own package** – You can use this type of model to create new model elements, and extend the metadata and business logic of referenced models. The wizard lets you select the referenced models. This type of model is compiled into its own assembly and binaries, and will simplify and reduce the cost of upgrades, deployment, and application lifecycle management in general.

- **A model that is a part of an existing package** – You can use this type of model to temporarily use legacy features such as overlayering source code and metadata. This feature is considered legacy and is supported only to upgrade from legacy versions.

In the Create model wizard, select *usr* for the layer. This layer will store user customizations. If needed, you can patch your customizations using the *usp* layer. If there are multiple versions of the same object in different layers, then the top layer will take precedence and will be used.

When the Create model wizard is completed, if you chose to create a new project, you will be prompted to specify a name and location for it.

Updating model parameters

If you must change the parameters for a model, you can use the Update model parameters dialog box.

1. On the Dynamics 365 menu, point to Model Management, and then click Update model parameters.
2. In the Model name field, select the model to update parameters for.
3. Update the parameters as you require.
4. Click Next.
5. Update the dependency information for the current model, if changes are required.
6. Click Next. The summary information for the model is displayed.
7. Click Finish.
The updated model parameters become effective only after you restart Visual Studio.

**Viewing package dependencies**

You can create a graphical representation that shows which packages and their models have dependencies on other packages. On the **Dynamics 365** menu, point to **Model Management**, and then click **View package dependencies**. A Directed Graph Markup Language (DGML) diagram will be generated for the current packages and their models. This diagram is a collection of interdependent nodes, each of which represents a package. Each node lists all the models that belong to that package. Additional tools let you enhance or simplify the diagram. For example, you can add comments, move nodes around, or remove nodes. You can also view package dependencies of a single model by following these steps:

1. Make sure the Application Explorer is in Model view: Right-click the AOT node and select **Model view**.
2. Right-click on any model and select **View package dependencies > View outgoing references**.

This will generate a graph of all packages that the selected model depends on.

**Deleting a model**

In a development or test environment, follow these steps to delete a model.

The following steps assume the local model store folder is `C:\AOSService\PackagesLocalDirectory` and your model is named `MyModel1`.

If your model belongs to its own package. For example an extension package with no other models in the package.

1. Close all instances of Visual Studio.
2. Stop the following services: AOS web service and Batch Management service.
3. Delete the package folder `C:\AOSService\PackagesLocalDirectory\MyModel1`. On cloud-hosted environments this folder may be located on another drive letter such as `K:`.
4. Restart the services that you stopped in step 1.


If your model belongs to a package with multiple models. For example, the MyModel1 overlays Application Suite.

1. Close all instances of Visual Studio.

2. Stop the following services: AOS web service and Batch Management service.

3. Delete the model folder C:\AOSService\PackagesLocalDirectory\<PackageName>\MyModel1. In this example, it's PackageName=ApplicationSuite. On cloud-hosted environments this folder may be located on another drive letter such as K:\.

4. Remove the descriptor file for the model in C:\AOSService\PackagesLocalDirectory\<PackageName>\Descriptor\MyModel1.xml. On cloud-hosted environments this folder may be located on another drive letter such as K:\.

5. Restart the services that you stopped in step 1.

6. In Visual Studio, build the package that the deleted models belonged to (Visual Studio > Dynamics 365 > Build models).


Additional resources

Development tools in Visual Studio

Develop and customize home page

Export and import models
The Finance and Operations project type is part of the development tools. This project type resembles other projects in Visual Studio. It helps you organize and manage the elements that you’re working with for a model. For example, the project can have folders that help you group the elements. A Visual Studio solution can contain multiple projects. There is one important constraint for a project: it can contain elements from only one model. If you must work with elements from different models, you must use multiple projects in your Visual Studio solution.

Create a new project

To create a new, empty project, follow these steps.

1. On the File menu, point to New, and then click Project.
2. In the list of template types, expand the Installed node.
3. Expand the Templates node.
4. Select the Finance and Operations category.
5. Select the Operations Project template.
6. Enter the name and location for the new project.
7. Specify whether you want to create a new solution or add the project to the current solution.
8. Click OK.

Every project has several important properties. To set the properties for a project, right-click the project in Solution Explorer, and then click Properties. The following table describes these properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Startup Object type</td>
<td>The type of object that will be used as the Startup Object when the project is run. The following types are available: Form, Class, Output menu item</td>
</tr>
<tr>
<td>Startup Object</td>
<td>The object that will be invoked when the project is run.</td>
</tr>
<tr>
<td>Company</td>
<td>The default company that will be used when the project is run.</td>
</tr>
<tr>
<td>Partition</td>
<td>The partition that will be used when the project is run.</td>
</tr>
<tr>
<td>Project File</td>
<td>The name of the file that contains information about the project.</td>
</tr>
<tr>
<td>Project Folder</td>
<td>The location of the project.</td>
</tr>
<tr>
<td>Model</td>
<td>The model that the project is associated with. All elements in the project must be in the selected model.</td>
</tr>
</tbody>
</table>
Of these properties, the **Model** property is particularly important. You must specify which model the project is associated with. All the elements that you create or add to the project must be part of this model. The **Startup Object type** and **Startup Object** properties are useful when you test and debug your application. When you start your project (by pressing F5 for debugging or Ctrl+F5 for no debugging), the specified form will be loaded, or the `main()` method from the specified class will be run. The method must have the following signature:
```java
public static void main(Args _args)
```

### Add elements to a project

There are several ways to add existing elements to a project. Here are the most typical:

- After you select the project in Solution Explorer, you can find the element in Application Explorer, right-click it, and then click **Add to Project**. This is the simplest method.
- You can use drag-and-drop operations to add an element from Application Explorer to a project.
- If you're limiting your search results to a single model, you can add the results of a filter in Application Explorer to a project.

After you've added the elements, you might want to use Solution Explorer to group them into folders, so that they are easier to find. The location of the project file and the folders that you create in the project don’t affect the location of the XML files that represent the model elements. The model elements are always stored in the appropriate folder in the model store. To organize elements into folders, select the **Organize projects by element type** option. On the **Dynamics 365** menu, click **Options**. Select the **Projects** category to see this option. When this option is selected, elements that are added to a new or existing project (such as when search results in Application Explorer are added to a project) are grouped into folders, based on the element type name.

To create a new element for a project, follow these steps.

1. In Solution Explorer, right-click the project, point to **Add**, and then click **New Item**.
2. In the **Operations Artifacts** list, select the category of element to create.
3. Select the specific element type.
4. Enter a name for the element.
5. Click **Add**. The element will be added to the project. It will also be added to the model in the model store that the project is associated with.

After you’ve added the new element, you might want to use Solution Explorer to move it into a folder in the project, so that it's easier to find.

### Export a projects as an .axpp file

To transfer elements to a different installation, you can use a project package file. Project package files have the .axpp file name extension. A project package contains all the elements from the project. To export a project, follow these steps.

1. In Solution Explorer, select the project to export.
2. On the Project menu, click Export Project. (The command on the menu will contain the name of the selected project.)

3. Enter a name for the project package file, and select a location.

4. Click Save.

Import an .axpp file

To use the contents of a project package file, you must import the .axpp file into an installation. The elements from the project package file will be imported into the same model that they were exported from. If that model doesn’t exist in the installation, it will be created during the import process. To import a project package file, follow these steps.

1. On the Dynamics 365 menu, click Import Project.

2. In the Import Project dialog box, specify the location of the project package (.axpp) file to import.

3. If you want elements from the project package file to overwrite any existing elements, select Overwrite Elements.

4. Specify whether you want to open the project in the current selection, in a new solution, or not at all.

5. In the Details field, review the elements that will be imported. You can clear the check box next to any elements that you don’t want to import.

6. Click OK to complete the import process.
This topic reviews the Add-ins infrastructure that has been added to Microsoft Visual Studio, so that developers can more easily add tools for development.

A lot of great tools have been added to Microsoft Visual Studio to support development. However, there will always be additional tools to meet specific requirements. To make it easier to add these additional tools, an Add-ins infrastructure has been provided for developers. The additional tools are available in two places:

- The Add-ins submenu on the Dynamics 365 menu
- The Add-ins submenu on the shortcut menu in the element designer

To make it easier to create your own add-ins, you can select the Dynamics Developer Tools Add-in project type when you create a new project in Visual Studio. This project type has the infrastructure that is required to implement an add-in.

For more information on add-ins, see:

- Visual Studio add-ins that support form patterns
This topic explains how to update the development tools.

Use this tutorial to update your Visual Studio development tools with a new version. It explains how to uninstall your existing Visual Studio development tools and install the new extension. The new extension is in the form of an installable VSIX file. This file is a part of the binary hotfix available on the Dynamics Lifecycle Services (LCS) site. The VSIX file is located in the `DevToolsService\Scripts` folder of the binary hotfix package.

**NOTE**
You do not need to follow the instructions in this article if you are upgrading your Finance and Operations platform to Platform update 4 or newer. It is an automatic step that is part of the platform upgrade process.

**Uninstall the existing Visual Studio extension**

In order to install a new version of the development tools, you’ll need to uninstall the existing version first. Verify the version of the development tools that you have installed. If you don’t have it installed, you can skip this section.

2. Select it and click *OK*.

**Uninstall the extension**

1. Open the Visual Studio *Tools > Extensions and Updates* dialog.
2. Select *Finance and Operations Visual Studio Tools* and click *Uninstall*.
3. When the extension is uninstalled, exit Visual Studio.

**Install a new version of the extension**

1. Make sure Visual Studio is not running.
2. Double-click (or right-click and *Open*) the VSIX file of the new version.
3. Follow the installation instructions.
4. When installation is complete, you can start Visual Studio and start developing your application.
This topic lists the Visual Studio components that are required to run the Visual Studio extension for X++.

NOTE
We do not recommend installing Visual Studio manually on the downloadable virtual hard drive (VHD) or virtual machines deployed from Lifecycle Services (LCS). Instead, we strongly recommend that you download or deploy a new virtual machine. Virtual machines deployed with versions 10.0.13 or later all have Visual Studio and its prerequisites installed, and come with other updates outside of Visual Studio to help keep the machines compatible and secure.

Visual Studio editions
The supported editions of Visual Studio include:

- Visual Studio 2017 Professional
- Visual Studio 2017 Enterprise
- Visual Studio 2019 Professional
- Visual Studio 2019 Enterprise

NOTE
Different Visual Studio editions have different licensing requirements and costs. For more information, see Visual Studio.

Required Visual Studio components
The following table lists the required Visual Studio components.

<table>
<thead>
<tr>
<th>TYPE</th>
<th>NAME</th>
<th>REQUIRED</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workload</td>
<td>.NET desktop development</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Individual component</td>
<td>Modeling SDK</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Individual component</td>
<td>DGML editor</td>
<td>No</td>
<td>This component is used for dependency graph features using the Directed Graph Markup Language.</td>
</tr>
<tr>
<td>Visual Studio Marketplace</td>
<td>Microsoft Reporting Services Projects</td>
<td>Yes</td>
<td>This component is needed for report development. If the component isn't installed, Visual Studio will prompt you when you try to open report designs.</td>
</tr>
</tbody>
</table>

Dynamics 365 Commerce
For more information on Commerce and Visual Studio, see these topics.
- Migrate the Retail SDK from Visual Studio 2015 to Visual Studio 2017
This topic reviews how you can debug X++ code by using the debugging feature in Microsoft Visual Studio.

To debug X++ code, you use the debugger in Microsoft Visual Studio. The process is similar to the process that is used for any other application that is created in Visual Studio. For example, the standard tools for examining the application are available when your code is stopped at a breakpoint.

### Debug your code

To debug X++ code, follow these steps.

1. In Visual Studio, open the X++ code to debug.

2. Find the line or lines where you want execution to stop, and set breakpoints in those lines. To set a breakpoint in a line, click in the left column of the code editor or press F9 while the cursor is on that line. A red dot indicates that a breakpoint has been set.

3. Set a startup project and a startup object. Startup objects can be any form, any class that has the main method, or any menu item. You can set the startup object in the Properties pane for the project. Alternatively, right-click the element in Solution Explorer, and then click Set as Startup Object.
4. On the **Debug** menu, click **Start Debugging**.

5. In the application, perform the action that causes the code that you're interested in to run. Typical actions include opening a form. Processing stops at the breakpoints that you set.

6. Use the tools in Visual Studio to examine the application. For example, you can hover over variables in the X++ code to see their values. You can also use commands on the **Debug** menu to step through the code. Additionally, tools such as the **Autos** pane in Visual Studio will show important information about the state of the application.
Another tool that is specific to Finance and Operations applications is the Infolog. Often, `info()` statements are added to code to log status messages while the application is running. You can view these Infolog messages directly in Visual Studio. On the `View` menu, click `Infolog`.

7. After you’ve finished debugging the application, exit the application. Visual Studio will exit debugging mode.

Add `ToString` methods to your classes

It’s often a benefit to add `ToString` methods to your classes. The effort spent doing this comes back many times and it’s easy to do.

```csharp
class MyClass
{
    public static void Main(Args a)
    {
        FMREntal rental;
        select EndMileage, RentalId from rental;
        rental.Comments = "Something";
    }
}
```

**NOTE**

Since `ToString` methods can be called at unpredictable times, it isn't a good idea to change the state of the object in the `ToString` method.

Identify unselected fields

It’s a common source of bugs to use fields from a table when those fields don’t appear in the field list in a `select` statement. Such fields will have a default value according to their type. You can use debugger breakpoints to determine if a value has been selected or not.

Consider the following code:

```csharp
Set a breakpoint on the assignment statement. Make your class the startup object in your project, and start by pressing F5. When the breakpoint is encountered, view the `rental` variable by expanding it in the `Locals` window. The fields that have been selected, `EndMileage` and `RentalId`, appear with their selected values, while the unselected fields appear as `null`. This means that their values weren't fetched from the database. Obviously, this is a debugging artifact. The values of the unselected fields will be the default value for the type of the field. Step over this and notice how the debugger changes the rendering to the actual value.
The Auto and Infolog windows

The debugger lets you easily access certain parts of the state of the application. This information is available in the Autos window, where the current company, the partition, the transaction level, and the current user ID are listed.

There is also a window showing the data that is written to the Infolog.

Breakpoint features

The Visual Studio debugger supports conditional breakpoints and breakpoints that are triggered by hit count. You can also have the system perform specific actions for you as you hit the breakpoint.

- **Hit count** enables you to determine how many times the breakpoint is hit before the debugger breaks execution. By default, the debugger breaks execution every time that the breakpoint is hit. You can set a hit count to tell the debugger to break every 2 times the breakpoint is hit, or every 10 times, or every 512 times, or any other number you choose. Hit counts can be useful because some bugs don’t appear the first time your program executes a loop, calls a function, or accesses a variable. Sometimes, the bug might not appear until the 100th or the 1000th iteration. To debug such a problem, you can set a breakpoint with a hit count of 100 or 1000.
- **Condition** is an expression that determines whether the breakpoint is hit or skipped. When the debugger reaches the breakpoint, it'll evaluate the condition. The breakpoint will be hit only if the condition is satisfied. You can use a condition with a location breakpoint to stop at a specified location only when a certain condition is true. For example, suppose you're debugging a banking program where the account balance is never allowed to go below zero. You might set breakpoints at certain locations in the code and attach a condition such as \( \text{balance} < 0 \) to each one. When you run the program, execution will break at those locations only when the balance is less than zero. You can examine variables and program state at the first breakpoint location, and then continue execution to the second breakpoint location, and so on.

- **Action** specifies something that should occur when the breakpoint is hit. By default, the debugger breaks execution, but you can choose to print a message or run a Visual Studio macro instead. If you decide to print a message instead of breaking, the breakpoint has an effect very similar to a Trace statement. This method of using breakpoints is called trace points.

### Using breakpoints with conditions

Consider the following code:

```csharp
class PVsClass
{
    public static void Main(Args a)
    {
        int i;
        for (i = 0; i < 10; i++)
        {
            print i;
        }
    }
}
```

Put a breakpoint on the print statements by pressing F9 while that statement is selected. This will create a normal, unconditional breakpoint. Now, use the mouse to open the context menu for the breakpoint and select **Condition**. Put in a condition that indicates that the breakpoint should happen when the value of the 'i' variable exceeds 5. Set the class as a startup project, and the class containing the code as the startup item in the project. Run the code. Feel free to modify the value of 'i' using the debugger. Now, remove this breakpoint, and use the Hit count feature to accomplish the same thing.

### NOTE

A breakpoint can have several conditions. It's often helpful to hover the cursor over the breakpoint, causing an informative tooltip to appear. Trace points are often useful to trace execution. Insert a trace point on the line in question and log the value of the variable. The trace output will appear in the output window in the debugger.

### The Immediate window

The **Immediate** window is a debugger feature that lets you enter expressions and statements to evaluate at any given time. This feature isn't implemented in the X++ stack. However, you can still benefit from the immediate window. The snippets must be expressed in C#, not in X++.
Delegate methods and delegate handler methods can be declared to support a request/response scenario, where the delegate calling logic requests the subscribers to provide a response. To support this scenario the `EventHandlerResult` class is most often passed as a parameter, and the delegate handler methods provide their result using one of the result methods on the class. However, the `EventHandlerResult` class can only contain a single result, so if multiple subscribers provide their individual result, the last respondent wins, and the results from the previous subscribers are overwritten.

Before the functionality described in this topic was introduced (platform update 5), there was no mechanism to ensure that, at most, a single subscriber provided a result, and that no results were lost if there were multiple subscribers.

### Ensuring, at most, one response

In platform update 5, the `EventHandlerResult` class has an additional static constructor which ensures that the logic fails if more than one subscriber provides a result. The new constructor is named `newSingleResponse`. When instantiating an `EventHandlerResult` object using this method, the framework will throw an exception as soon as a second delegate handler method attempts to provide a result.

```cpp
EventHandlerResult result = EventHandlerResult::newSingleResponse();
this.validateWarehouseTypeDelegate(this.WarehouseType, result);
```

### IEventHandlerResultValidator interface

The validation in the `EventHandlerResult` class is handled by injecting an object of a type that implements the `IEventHandlerResultValidator` interface. When instantiating the `EventHandlerResult` object using the `newSingleResponse` static constructor, an `EventHandlerSingleResponseValidator` object is instantiated and injected into the `EventHandlerResult` object, and the injected object becomes responsible for validating any result provided to the `EventHandlerResult` object. Other validation classes can be implemented by having the class implement the `IEventHandlerResultValidator` interface, and injecting it into the `EventHandlerResult` class by instantiating the `EventHandlerResult` object using another new static constructor named `newWithResultValidator`. The constructor takes an argument of type `IEventHandlerResultValidator`, which makes it possible to inject any validator object as long as it implements the `IEventHandlerResultValidator` interface.

For example, the `newSingleResponse` static constructor simply delegates the instantiation to the `newWithResultValidator` static constructor like this.

```cpp
return EventHandlerResult::newWithResultValidator(EventHandlerSingleResponseValidator::construct());
```

### Accept and reject request/response scenarios

In certain request/response scenarios, the subscriber is only expected to provide their acceptance or rejection. Using the `EventHandlerResult` class to request acceptance/rejection can be confusing, if the subscriber is only
expected to respond with a Boolean value. In a validation scenario, for example, should the subscriber only respond with Boolean false, when validation fails, or should the subscriber also respond with Boolean true, if validation succeeds? If the response is gathered using an **EventHandlerResult** object, then the second subscriber that validates and replies with Boolean true, might overwrite the Boolean false from the first subscriber.

To mitigate this confusion, two new result type classes have been introduced in Platform update 5: **EventHandlerAcceptResult** and **EventHandlerRejectResult**.

When using the **EventHandlerAcceptResult** class, the delegate handler method can only respond by calling the **accept** method. When using the **EventHandlerRejectResult** class, only the **reject** method can be called.

```java
public static void validateWarehouseTypeIsSupportedStandardDelegateHandler(
    InventLocationType _inventLocationType,
    EventHandlerAcceptResult _result)
{
    switch (_inventLocationType)
    {
        case InventLocationType::Standard:
        case InventLocationType::Quarantine:
        case InventLocationType::Transit:
            _result.accept();
            break;
    }
}
```

The two new classes also contain a **newSingleResponse** static constructor for use in scenarios where, at most, one subscriber is allowed to respond with their rejection or acceptance. Whether any subscriber has responded can still be answered by querying the **hasResult** method, and the acceptance/rejection is queried by calling either the **isAccepted** or **isRejected** methods for the **EventHandlerAcceptResult** and **EventHandlerRejectResult** classes, respectively.

```java
boolean ret = false;
EventHandlerAcceptResult result = EventHandlerAcceptResult::newSingleResponse();
this.validateWarehouseTypeDelegate(this.WarehouseType, result);
if (result.hasResult())
{
    ret = result.isAccepted();
}
```
The primary goal of this tutorial is to illustrate the interoperability between C# and X++. In this tutorial, you’ll write business logic in C# source code and in X++ source code.

In this tutorial, you’ll write business logic in C# source code and in X++ source code. You’ll get experience with the following:

- New tools in Visual Studio.
- The handling of events in C#.
- The use of Language Integrated Query (LINQ) in C# to fetch data.

**PrEREQUISITE**

This tutorial requires that you access the environment using Remote Desktop, and be provisioned as an administrator on the instance.

**NOTE**

Debugging support for the C# project does not work if the **Load symbols only for items in the solution** check box is selected. Since this option is selected by default, it must be changed prior to running the lab. In Visual Studio, click **Dynamics 365 > Options**, and clear the **Load symbols only for items in the solution** check box.

**Scenario**

Too many cars have been rented to drivers who have a history of unsafe driving habits. The Fleet Management rental company needs to check driving records from external sources. Upper management has decided to subscribe to a service that is hosted by the Department of Transportation (DOT), which is the legal entity that manages drivers’ licenses and associated information. This service retrieves the number of citations for the given unique license number. It’s not easy to call external services directly from X++ source code. Visual Studio has tools for generating the “code-behind” (in C#) that calls the services, and these tools make the development effort easy. The obvious choice would be to leverage Visual Studio to write the code. However, in this tutorial your code won’t actually call an external service, because the logistics are beyond the scope of the simple lab environment. Instead, we provide a mock implementation of a service call. The goal of this tutorial is to teach an understanding of the current state of C# and of interoperability with X++.

**Create a C# class library**

You can create a reference from a project to the C# class library, or to any other type of C# project that generates an assembly. Such references affect the build order. The C# project is built before the project that references and depends on it. The infrastructure understands the references, and will make sure that the C# assemblies are deployed correctly to the cloud before execution. Follow these steps to create a C# class library in the Fleet Management solution:

1. In Visual Studio, click **File > Open project/solution**.

2. In the **Open Project** dialog box, in the **File name** text box, type the following path, and then press
Enter: C:\users\public\desktop\FleetManagement

3. Select the file named FleetManagement.sln, and then click Open. If the solution file is not on your computer, the steps to create it are listed in Tutorial: Create a Fleet Management solution file out of the Fleet Management models in the AOT.

![Open Project Dialog]

4. Right-click the FleetManagement solution, and then click Add > New Project. The Add New Project dialog is displayed.

5. In the left pane, click Visual C#, and then in the middle pane, click Class Library.

6. At the bottom in the Name text box, type the name DriversLicenseEvaluator.

7. In the Location text box, type the following directory path: C:\users\public\desktop\FleetManagement.

8. Verify that your project is set to "NET Framework 4.5" in the drop-down list at the top.

9. Click OK to create the project.

![Add New Project Dialog]

10. In Solution Explorer, under the DriversLicenseEvaluator project, right-click the file name Class1.cs and rename it DriversLicenseChecker.cs.
Write a C# method named CheckDriversLicense

In this section, you add C# code for a method named CheckDriversLicense. The method must validate the driver’s license. To do this, the method must retrieve the driver’s license number, which is stored in the customer table. The method is given the Recl value for the customer record that contains the information required by the method. Your C# code uses the LINQ provider to read from the customer table. For LINQ to work, you must first add references pointing to the LINQ assemblies. You add these references to the C# project named DriversLicenseEvaluator.

1. In Solution Explorer, expand the DriversLicenseEvaluator project node, right-click References, and then click Add Reference.

2. Click Browse and then enter the following path: C:\Packages\bin

   Note that in some environments, the location of the packages folder is not on the c: drive.

3. In the File name field, type the pattern "*LINQ*.dll" and then press Enter. You’ll see a list of assemblies with the name LINQ in them. From that list, select the following files, and then click Add:

   - Microsoft.Dynamics.AX.Framework.Linq.Data.dll
4. You must also add the support assemblies that contain the Common type that you'll use in the code below. Click **Browse** again, and then type the following file name into the field:

- Microsoft.Dynamics.AX.Xpp.Support.dll
- Microsoft.Dynamics.AX.Data.Core.dll

5. Click **Add**, and then click **OK**. The assemblies now appear under the references node in the project.

6. Repeat the **Add Reference** process, except this time, add the following DLL file from the indicated path:

- Dynamics.Ax.FleetManagement.dll, in C:\Packages\FleetManagement\bin

7. In **Solution Explorer**, select the reference Dynamics.Ax.FleetManagement.dll reference and set the property **Copy Local** = **False**.

8. In **Solution Explorer**, right-click **DriversLicenseChecker.cs**, and then click **View Code**.

9. Add the following three using statements to the **DriversLicenseEvaluator** namespace, to reduce the verbosity of code that references external classes. using Dynamics.AX.Application; using Microsoft.Dynamics.AX.Framework.Linq.Data; using Microsoft.Dynamics.AX.Xpp; Your C# code should now look something like the following example.

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DriversLicenseEvaluator
{
    using Dynamics.AX.Application;
    using Microsoft.Dynamics.AX.Framework.Linq.Data;
    using Microsoft.Dynamics.AX.Xpp;

    public class DriversLicenseChecker
    {
    }
}
```

10. Replace the class CheckDriversLicense with the following code.

    **TIP**

    If you prefer, you can paste in the code from the DriversLicenseChecker.cs file in the C:\FMLab directory.
public class DriversLicenseChecker
{
    public static bool CheckDriversLicense(long customerId)
    {
        // Use LINQ to get back to the information about the license number
        FMCustomer customer;
        QueryProvider provider = new AXQueryProvider(null);
        var customers = new QueryCollection<FMCustomer>(provider);

        // Build the query (but do not execute it)
        var query = from c in customers
                    where c.RecId == customerId
                    select c;

        // Execute the query:
        customer = query.FirstOrDefault();
        if (customer == null)
        {
            throw new ArgumentException
            ("The customerId does not designate a customer");
        }

        if (string.IsNullOrEmpty(customer.DriverLicense))
        {
            // No driver's license was recorded. Veto the rental.
            return false;
        }

        // Call the DOT web service to validate the license number.
        // This is not practical for this lab, because all the service providers
        // charge for this service. Instead, just assume that any license number
        // that contains the sequence "89" is valid.
        // In the demo data, this is true for Adrian Lannin,
        // but not for Phil Spencer.
        return customer.DriverLicense.Contains("89");
    }
}

Understand the LINQ code

Before proceeding with more C# code, verify that you understand the LINQ code you just added. More details about LINQ are provided in the Technical Concepts Guide, so only the basics are described below.

- First, a provider is created. It provides access to all the tables.
- Next, a collection of all customers is created. The customer of interest is retrieved from this collection.
- Then, a query is created with a where clause that designates the requested customer by RecId.
- The call to the FirstOrDefault method forces execution of the query.
- The method assigns the single matching customer to the customer variable. (Null is assigned if the RecId value matches no customer)
- Finally, the customer data is tested to see if the associated driver's license is valid. (Does the license contain "89"?)

Handle the event when a record is added

The following subsections provide the following:

- Explain the upcoming code items and their inter-relationships.
- Show the code for an event handler.
- Associate the handler with the event occurrences.

Preparatory overview
When an attempt is made to add a record to a table, the OnValidateWrite event is raised before the record is written to the database. You want your CheckDriversLicense method to be called each time on the OnValidateWrite event is raised for the FMRental table. To do this, you now need to write a C# method that is invoked by the event, and which calls your checkDriversLicense method. In other words, you need to write an event handler that calls your CheckDriversLicense method. The event handler method receives a parameter of the type, DataEventArgs. The event handler can set a value in the DataEventArgs structure to accept or reject the record. After you write your event handler method, you connect it to the event by assigning, or adding it to the OnValidatedWrite delegate that is a member of the FMRental table. You write this assignment in the init method of the data source of the FMRental form. This assignment to a delegate might seem odd. After all, we're modifying existing code (FMRental) to add handlers, which contradicts the main value proposition of loose coupling that eventing is supposed to offer. This assignment step is temporary. We'll eventually have the same story in C# as we do in X++, where an attribute is applied to the C# event handler as the mechanism that ties the delegate to the handler.

### Write an event handler method

In C#, write the following event handler method and add it to the DriversLicenseChecker class.

```csharp
public static void OnValidatedWriteHandler(Common table, DataEventArgs args)
{
    var validateEventArgs = args as ValidateEventArgs;

    // Do not check if already rejected.
    if (validateEventArgs.parmValidateResult())
    {
        var rentalTable = table as FMRental;
        if (rentalTable == null)
        {
            throw new ArgumentNullException("table");
        }

        var result = CheckDriversLicense(rentalTable.Customer);
        validateEventArgs.parmValidateResult(result);
    }
}
```

Build the DriversLicenseEvaluator project by right-clicking the project node and then clicking **Build**.

### Add a reference pointing to the DriversLicenseEvaluator project

Create a reference from the X++ project named **FleetManagement Migrated** to the C# project named **DriversLicenseEvaluator**, by completing the following steps.

1. Right-click the FleetManagement Migrated project, click **Add**, and then click **Reference**. Select the row for the DriversLicenseEvaluator project in the **Projects** references tab, and then click **OK**.
2. Under the FleetManagement Migrated project, expand the References node, and there you see new reference to the DriversLicenseEvaluator project.

Build sequence

Your C# DriversLicenseEvaluator project will be built before the FleetManagement Migrated project is built. This is because the added reference makes the Fleet project dependent on your project. The build sequence is easy to see if you right-click the FleetManagement solution, click Project Build Order, and then click Dependencies.
Add your event handler to a delegate

1. In Solution Explorer, navigate to FleetManagement Migrated > User Interface > Forms > FMRental.

2. Double-click the FMRental form. The Visual Studio designer opens to the form.

3. Expand the Data Sources node to show the data sources used in the form.

4. Expand the FMRental data source, and then the Methods node to list the methods defined on the data source.

5. Right-click Methods, and then click Override > init. The list displays all of the methods on the data source that haven't yet been overridden. When you select init, this opens the file FMRental.xpp in the X++ code editor with the cursor near the template for the init method.

6. At the end of the init method body, use the += operator to add one assignment to a delegate.

   ```
   FMRental.onValidatedWrite += eventhandler
   (DriversLicenseEvaluator.DriversLicenseChecker::OnValidatedWriteHandler);
   ```

7. Click to save, and then build the entire solution.

Final test

In this section, you set breakpoints and run the Fleet application under the Visual Studio debugger. This enables you to prove the following:

- Your LINQ query runs when the OnValidateWrite event is raised.
- Your LINQ query successfully retrieves the data for one customer.

Prepare the test

1. In Solution Explorer, navigate to FleetManagement Migrated > User Interface > Forms.

2. Right-click FMRental, and then click Set as Startup Object.

3. In the code editor for DriversLicenseChecker.cs, find the OnValidateWriteHandler method. Find the following line of code.
4. Set a breakpoint on that line of code. You do this by clicking in the left margin at that line. A red dot displays when the breakpoint is set.

5. In the CheckDriversLicense method, set another breakpoint at the following line.

```csharp
if (string.IsNullOrEmpty(customer.DriverLicense))
```

**Run the test**

For this test, we'll be debugging the C# code that we've written. To do this, we need to inform Visual Studio to load the symbols for the assembly that contains the C# code. Go to Dynamics 365 > Options > Debugging and verify that the Load symbols only for items in the solution check box is not selected.

![Options window](image)

**TIP**

If you’re unable to get to the breakpoint in the C# code, you may want to open the Modules window (Debug > Windows > Modules), find the C# module and load it explicitly.

1. Click Debug > Start Debugging. This starts the Fleet application, and a browser window with the FM Rental form is displayed.

2. Click on any Vehicle rental ID to view details.

3. Click the Edit icon near the top left of the form. The icon looks like a pencil.

4. In the To field of the Rental section, increase the date by one day.

5. Click the Save button. This causes the focus to shift to Visual Studio at your highlighted breakpoint. This line shows that the OnValidatedWrite event was raised, and that your handler method was called.

6. Press F5 to continue the run. Instantly, your other breakpoint becomes highlighted.

7. Find the variable customer a few lines above your breakpoint.

8. Right-click the customer variable, and then click QuickWatch. Any long integer value proves that your
LINQ query worked.

9. Press F5 to complete the Save operation.
This topic discusses the LINQ provider.

LINQ (Language Integrated Query) is a set of classes and methods that enable you to access data that is stored in a variety of places and formats. The LINQ framework is the standard for accessing data in managed languages. LINQ presents to programmers a unified and consistent API for data access from heterogeneous data sources, such as:

- In-memory object graphs
- Active Directory entries
- Flickr pictures and XML
- SQL Server

The LINQ provider allows the user to access business data by using .NET managed languages.

Two syntactical mechanisms for accessing LINQ

There are two syntactical approaches for using LINQ, as described in the following table.

<table>
<thead>
<tr>
<th>APPROACH</th>
<th>X++</th>
<th>C# AND VISUAL BASIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINQ by standard method call syntax.</td>
<td>Impractical. Language support for generics is vital for LINQ and is not supported in X++.</td>
<td>Available, requires lambda syntax.</td>
</tr>
<tr>
<td>LINQ by specialized syntax that is understood by the compiler.</td>
<td>Not available.</td>
<td>Available, easier to use.</td>
</tr>
</tbody>
</table>

There are two syntactic mechanisms for accessing the LINQ provider in C# (or in Visual Basic):

- By standard, or fluent, method call syntax.
- By specialized syntax that the C# compiler has been enhanced to understand as equivalent to the LINQ method calls. (Such syntax is sometimes called "syntactic sugar").

This topic is going to review each syntactic mechanism for LINQ, starting with the easier specialized syntax.

LINQ by specialized syntax in C#

Some .NET languages understand specialized syntax for LINQ as an alternative that is easier for us to write. C# is one such language.

To use LINQ in C#, you must understand the C# keyword `var`, which is used to declare variables. The `var` keyword tells the compiler to figure out the data type of the variable by what is assigned to the variable. This feature is now also available in X++. The type is implicit in the source code, and the type is settled and unvarying after the compilation completes.

Comparing X++ to C# LINQ

The X++ language supports the useful and easy to use `while select` statement. This lets you compare the X++ `while select` syntax to the specialized C# LINQ syntax. First, here is the X++ sample.
CustTable ct;    // X++, traditional while select.
CustTrans trans;

while select * from ct
    where ct.AccountNum >= '4000'
    join RecId from trans
    where trans.RecId == ct.RecId
    order by ct.AccountNum desc
{
    print ct.AccountNum;
}

Next is an equivalent query in C# with the specialized LINQ syntax.

    // C#, with specialized LINQ syntax.
    // Get access to the data provider:
    var provider = new QueryProvider(null);

    var customers = new QueryCollection(provider);
    var customerTransactions = new QueryCollection(provider);

    var query = from ct in customers
                  from trans in customerTransactions
                  where ct.AccountNum >= "4000"
                  where trans.AccountNum == ct.AccountNum
                  order by ct.AccountNum descending
                  select ct;

    foreach (var ct in query)
    {
        System.Console.WriteLine(ct.AccountNum);
    }

LINQ query in C# by method syntax, using the lambda operator >

Next is another use of LINQ in C#, except this time the more standard syntax is used to call the LINQ API. The approach also involves use of the lambda operator >. The following C# query is functionally equivalent to the preceding C# query.

    var query = customers
                .Where(c => string.Compare(c.AccountNum, "4000") >= 0)
                .Join(customers,
                      primary => primary.AccountNum,
                      foreign => foreign.AccountNum,
                      (primary, foreign) => new { P = primary, F = foreign })
                .OrderBy(primaryAndForeign => primaryAndForeign.P.AccountNum)
                .Select(primaryAndForeign => primaryAndForeign.P);

There's a good match between the while select syntax used in X++ and the specialized LINQ syntax in C# (Visual Basic has particularly good LINQ syntax). It's evident that the specialized LINQ syntax is actually very useful in expressing joins, but the specialized syntax built into the C# compiler doesn't handle the extensions provided by the Finance and Operations applications.

Limitation of the specialized LINQ syntax

A limitation of the specialized LINQ syntax is that it can't be augmented with extensions to the LINQ provider. In contrast, standard syntax of method calls plus the lambda operator can be extended as needed. For instance, the LINQ framework provides a method for cross-company hints that can't be expressed in the special syntax for LINQ in C#. Fortunately, due to the ability to compose queries, this limitation need not be a major problem. Calls
LINQ query execution

The code generated for a LINQ query builds a tree at run time. When the results of the query are required, this tree is passed to the backend that will interpret it, and will provide the data as expressed in the query. The X++ compiler also builds a tree to express the query, but the X++ compiler has intimate knowledge about the capabilities of the database backend. This has several important implications as described in the following subsections.

Inability to diagnose problems at LINQ compile-time

The C# compiler is largely unable to foresee and diagnose errors that will occur at run time due to the inability of the backend to process an incompatible LINQ query. For instance, in the following C# code block, the specialized LINQ syntax is valid according to the C# compiler. Yet at run time, an error would occur.

```csharp
var customerQuery = from c in db.Customers
                      where (from o in db.Orders
                              where o.ShipCountry == "Germany"
                              select o.CustomerID).Contains(c.CustomerID);
```

This query can't be handled by the current data layer, and while no errors are diagnosed at compile time, an error would occur at run time.

Performance penalty with LINQ

There is an overhead penalty paid at run time, when analysis of the tree occurs, and a suitable access language is generated. As we might expect, the performance penalty is incurred when analyzing the LINQ expression tree. The time required at run time to actually fetch the data doesn't vary much between C# and LINQ versus X++ with while select. Our preliminary numbers show that the beginning-to-end performance of the query is about three times longer with C# LINQ compared to X++ while select, when very few records are fetched. But when many records are fetched, the total times are about the same between C# and X++. The conclusion is that it takes much longer to fetch a lot of records than it takes to analyze the language tree.

Composing queries with LINQ

The model that's provided by LINQ allows queries to be composed of subqueries. The X++ language can't cleanly provide this feature. To understand this, consider the following C# LINQ code. A flag is passed to a method to control the ordering of data results.
private IEnumerable RichCustomers(bool orderByName)
{
    // Create a query for the rich customers. Note carefully
    // that no data is fetched when this is executed.
    var q = from c in customers where c.AmountMst > 1000000.0m select c;

    if (orderByName)
    {
        // Add the order by clause to the existing query.
        // Still no data is yet fetched.
        return q.OrderBy(c => c.AccountNum);
    }
    else
    {
        return q;
    }
}

Set based operations with LINQ
LINQ queries can be applied for CRUD operations. But the model for updating, deleting, and inserting records
isn't useful for the expression of set based operations. We're now working on extensions to add to the LINQ
model that will translate into set based operations.
This topic describes how you can author best practice rules in C#, for both metadata and X++ code. Best practice checks are run by the compiler. You can run them in daily builds to catch objectionable practices that are unacceptable in shipping code. The features can also be used to author simple one-of tools to gather information about the application.

You can also use best practice rules to author simple tools that gather information about your application. The framework is built on top of a managed framework called XLNT (shorthand for X++ LaNguaGe Toolkit). You can use the framework to build custom tools that extract information from, and modify, X++ code. There are two types of best practice rules: rules that deal with metadata and rules that deal with source code.

**Code Best Practice framework**

The Code Best Practice Framework (CBPF) enables you to write your own tools for analyzing X++ source code. These rules diagnose things that you consider to be problems with X++ source code. This section describes the foundation of the Best Practice functionality. This information is helpful for understanding the later sections that describe creating your own rules in greater detail. It is also helpful to developers who want to code rules that are more complex than those demonstrated in this document. The CBPF API lets you focus on the rule you are expressing, without having to deal with infrastructure issues. You don't need to read tokens and piece them together to create something intelligible from them. Instead, the CBPF provides the following parts:

- A parser that builds an Abstract Syntax Tree (AST) from X++ source code.
- A pipeline that runs a sequence of passes over the X++ code.
- A number of prebuilt passes. The first pass is the parsing of the source code.
- Infrastructure to read metadata.

Because rules are based on ASTs, it is important to understand that concept before starting to write rules.

**The parser and ASTs**

The parser reads X++ code and produces an AST from it if it does not contain egregious syntax errors. The parser has a built-in error recovery scheme, so it can recover from most syntax errors reasonably well. When a syntax error happens, the parser will read symbols until it encounters a semicolon, and then try to building the AST by unstacking its state until it reaches a state where a semicolon is a correct symbol. In addition, the parser is able to suggest the correct set of symbols when a syntax error occurs. The parser is not intended to be directly interacted with by consumers of the API, but should be considered a black box that works without user intervention. As the parser recognizes the language constructs in the source code, it builds an AST. The AST consists of nodes that are abstractions of the X++ artifact they represent. We will illustrate the concept by showing a few AST nodes below:

```csharp
public abstract class Statement : Ast
{
    /// <summary>
    /// Gets or sets the comments in the statement
    /// </summary>
    public string Comments { get; set; }

    public abstract string ToString(int indent);
}
```
The **Statement** class is abstract, because it doesn't make sense to instantiate a "statement". Only concrete derived classes, like **if** and **while** statements, can be instantiated. Since the Comments property is placed on this base class, it applies to all derived classes. In other words, all statements can have comments preceding the statement. The comments are accessible through the given property. There are many different kinds of statement in X++ and each one is described by a class derived in one or more steps from the abstract **Statement** class shown above. The following example shows the definition of a **while** statement.

```csharp
public class WhileStatement : Statement
{
    /// <summary>
    /// Gets or sets the while condition.
    /// </summary>
    public Expression Condition { get; set; }

    /// <summary>
    /// Gets or sets the constituent while statement.
    /// </summary>
    public Statement Statement { get; set; }
}
```

The **while** statement consists of the condition (an expression) and the constituent statement, that is executed as long as the condition evaluates to true. The parser will maintain the source code positions where the represented artifact starts and ends (that is, its extent). As the ASTs are traversed, it may be useful to add information to the individual nodes. For example, every expression has a type. As the tree is traversed to diagnose type compatibility problems it becomes useful to be able to place that information on the individual node. Rather than having to modify the AST nodes for each requirement, there is a property collection that can be used to provide name/value pairs to each node. Each AST node has a **ToString** method that will return a high fidelity string representation of the node, which is useful in debugging scenarios.

**The AstSweeper class**

The AstSweeper applies a visitor pattern to the AST instance that it is given. The visitor pattern allows the programmer to separate the underlying data structure (that is, the AST) from the operations that the user wants to perform on the nodes (that is, the logic reasoning about the code). The **AstSweeper** class has a virtual method for each of the AST node types, and it will call them as directed by the structure of the AST. The following examples show how the sweater works. Some details have been omitted for clarity.
The name of the virtual method handling a particular AST node is the name of the AST class prepended with Visit. The parameters are the node to visit and a payload that may be passed to all the visitors as they are called. In this way, the sweeper will call the virtual method once for each and every one of the nodes in the AST that is passed to it in a depth-first traversal. The payload parameter can be used to pass information (for example, a symbol table) to each node as required. Developers will build classes derived from the AstSweeper class, overriding the methods of particular interest to them.

**Example**

Suppose you need to determine the percentage of parameter names starting with an underscore character, thus conforming to the X++ coding guidelines. You would then write a class deriving from the AstSweeper class with logic that calculated the number of parameters and the number of parameters starting with underscore. The following example shows this code.
public class ParameterCounter : AstSweeper<object, object>
{
    /// <summary>
    /// The parameter count maintained as the methods are encountered.
    /// </summary>
    public int ParameterCount { get; set; }

    /// <summary>
    /// The number of parameters that start with an underscore character.
    /// </summary>
    public int UnderscoredParameters { get; set; }

    /// <summary>
    /// Visits the method parameters.
    /// </summary>
    /// <param name="o">The payload. Not used in this scenario.</param>
    /// <param name="parameters">The list of parameters to visit..</param>
    /// <returns>The method parameters.</returns>
    protected override object VisitMethodParameters(object o, IEnumerable<ParameterDeclaration> parameters)
    {
        this.ParameterCount += parameters.Count();
        this.UnderscoredParameters += parameters
            .Where(p => p.Name.StartsWith("_"));

        return null;
    }
}

In this case, the tally is maintained in the **ParametersCount** and **UnderscoredParameters** properties that are defined in the class scope. Another equally valid approach would be to pass this information into the payload that is passed to all the **Visit** methods. In most cases, the user should unconditionally call **super()** from the overridden method to make sure that the **Visit** methods are called for all nodes below the one being visited. In the case above, it does not make a difference so we opt to improve performance by pruning the AST tree traversal.

**Writing code for Best Practice rules**

To author a business rule:

1. Define the situation you want to diagnose in terms of properties of the AST. You will write **Visit** methods that can do the analysis.
2. When the error condition has been found, a diagnostic message must be generated. There is an API that is used for this purpose; basically you need to write some boilerplate code for each diagnostic message.
3. You need to hook your new best practice rule into the rest of the framework, so the user can decide whether or not to include your rule and to run it if so directed.

**Create a best practice rules project in Visual Studio**

In this walkthrough, we imagine the following scenario:

- Some methods are adorned with an **Author** attribute that provides the name of the individual who wrote the code. The attribute is useful when finding who to point the finger at when stack traces containing that method appear.
- Since we have a significant turnaround of developers, the names of the developers listed cannot be static. We want to find the names that are used in the **Author** attributes, and match them against a list of names of current developers.

The author attribute class is defined as:
class AuthorAttribute extends SysAttribute
{
    private str theAuthor;

    public void new(str _author)
    {
        this.theAuthor = _author;
    }
}

In production code, we would put in documentation comments and assertions to validate key assumptions about parameter values etc. For the sake of clarity, we omit these steps in this walkthrough, where the code is written for clarity. Now that we have set the stage for what we want to achieve, we can start up Visual Studio and create a best practice rules project. Provide a meaningful name that properly conveys what the rules are intended to do: Visual Studio creates a project with some source code snippets and project references set up. You can save considerable time by using this source code as a starting point for your own code. The pre-canned example contains rules that prohibit the word “Microsoft” in any method names (presumably for copyright reasons) and a metadata-based rule prohibiting certain characters in names. Since we are not concerned with the metadata checks for now, you can delete the InvalidCharactersDiagnosticItem.cs and DemoMetadataCheck.cs files from the project. Also, since we are not interested in the Microsoft name check, go ahead and delete the content of the VisitMethod method in the DemoAST class. The first thing we need to do is to find out if there are one or more Author attributes for a particular method. You will notice that the Method type (that is passed as a parameter to the VisitMethod method) has an Attributes property of type AttributeList. Let’s use it to see if any Author attributes are defined on this method:

protected override object VisitMethod(BestPracticeCheckerPayload payload, Method method)
{
    var names = new List<string>();
    foreach (var attribute in method.Attributes.Attributes)
    {
        if (string.Compare(attribute.Name, "Author", ignoreCase: true, culture: CultureInfo.InvariantCulture) == 0)
        {
            var authorNameLiteral = attribute.Parameters.First().Literal as StringAttributeLiteral;
            // The name contains quotes (either single or double). Get rid of those
            var authorName = authorNameLiteral.Value.Trim('\', '\');
            names.Add(authorName);
        }
    }

    // More to come...
    return null;
}

At this point we have looped through any attributes, and collected a list of author names, that is names that are provided as the first parameters to the Author attributes. Now we need to compare the list against a list of acceptable authors, that we maintain in a static list. Whenever an author is provided that is not mentioned in the list we need to issue an appropriate diagnostic message. At this time, we have something like:
public class AuthorListRule : BestPracticeAstChecker<BestPracticeCheckerPayload>
{
    private static HashSet<string> authorlist = new HashSet<string>()
    {
    }

    public AuthorListRule() : base()
    {
    }

    protected override object VisitMethod(BestPracticeCheckerPayload payload, Method method)
    {
        var names = new List<string>();
        foreach (var attribute in method.Attributes.Attributes)
        {
            if (string.Compare(attribute.Name, "Author",
                                ignoreCase: true, culture: CultureInfo.InvariantCulture) == 0)
            {
                var authorNameLiteral = attribute.Parameters.First().Literal as StringAttributeLiteral;
                // The name contains quotes (either single or double). Get rid of those
                var authorName = authorNameLiteral.Value.Trim(\"", \"\");
                names.Add(authorName);
            }
        }

        foreach (var name in names)
        {
            if (!authorlist.Contains(name))
            {
                // TODO: Add a diagnostic message
            }
        }
        return null;
    }
}

In other words, we need to create a diagnostic message to let the user know about the transgression of the rule. As noted before, it is important to call the base implementation of your visitor, which will then call visitor methods for all the nodes that are contained in the method. However, in this case, we do not want to do any further processing once we have determined if the author attribute is on the list.

**Add a class for the diagnostic message**

The project already includes boilerplate code for an error message, so we will use that as our starting point to create the diagnostic message that will be returned if the rule is violated. Each message is implemented as a class of its own. Each error message may have any amount of contextual information encoded into it. In this case, the contextual information is the name of the author that is not found in the list. We will start by adding the message to the messages resource file: Open that file in the project and add a string to it. We will use the name (also known as the error moniker) AuthorNotCurrent. The \'{0}\' string is a placeholder for the contextual information, in this case the name of the author who is not in the list. In addition to the actual text that will appear in the error message, there is also a string containing a description of the rule. This information is shown in the best practice dialog within Visual Studio and is designed to help the user decide which rules to enable on the system. Create a class for the diagnostic message, and call it `AuthorNotCurrentDiagnosticItem.cs`. Add the following code inspired from the `NotAllowedWordDiagnosticItem.cs` class.
namespace CompareAuthorsToList
{
    using System;
    using System.Collections.Generic;
    using System.Runtime.Serialization;
    using System.Xml.Linq;
    using Microsoft.Dynamics.AX.Metadata.XppCompiler;

    [DataContract]
    public class AuthorNotCurrentDiagnosticItem : CustomDiagnosticItem
    
    private const string AuthorNotCurrentKey = "Author";
    public const string DiagnosticMoniker = "AuthorNotCurrent";

    public AuthorNotCurrentDiagnosticItem(string path, string elementType, TextPosition textPosition, string author)
    : base(path, elementType, textPosition, DiagnosticType.BestPractices, Severity.Warning, DiagnosticMoniker, Messages.AuthorNotCurrent, author)
    
    {this.AuthorNotCurrent = author;}

    public AuthorNotCurrentDiagnosticItem(Stack<Ast> context, TextPosition textPosition, string author)
    : base(context, textPosition, DiagnosticType.BestPractices, Severity.Warning, DiagnosticMoniker, Messages.AuthorNotCurrent, author)
    
    {this.AuthorNotCurrent = author;}

    // Serialization support
    public AuthorNotCurrentDiagnosticItem(XElement element)
    : base(element)
    
    { }

    [DataMember]
    public string AuthorNotCurrent { get; private set; }

    /// <summary>
    /// Hydrate the diagnostic item from the given XML element.
    /// </summary>
    /// <param name="itemSpecificNode">The XML element containing the diagnostic.</param>
    protected override void ReadItemSpecificFields(System.Xml.Linq.XElement itemSpecificNode)
    
    {this.AuthorNotCurrent = base.ReadCustomField(itemSpecificNode, AuthorNotCurrentKey);}

    /// <summary>
    /// Write the state into the given XML element.
    /// </summary>
    /// <param name="itemSpecificNode">The element into which the state is persisted.</param>
    protected override void WriteItemSpecificFields(System.Xml.Linq.XElement itemSpecificNode)
    
    {this.WriteCustomField(itemSpecificNode, AuthorNotCurrentKey, this.AuthorNotCurrent);}
}
As you can see, there are four parameters specified for the **BestPracticeRule** attribute:

1. The rule moniker.
2. The type of the resource file holding the rule description. In this example, we are using the default resource file named `Messages`, which created a class called `Messages`. We want the type of this class as the second argument.
3. The name of the string resource that contains the description of the rule. This name is the string called `AuthorNotCurrentDescription` that we added to the resource file above; it contains a human legible string to describe the rule. This string is used to describe the rule to the user in a best practice dialog within Visual Studio. In Visual Studio, select `Dynamics 365 > Best Practices Configuration` to view the dialog.
4. A description of the artifacts to check. In this case, the value specifies that the rule should only be applied to classes. Modify the description as needed by using one of the other literals in the `BestPracticeCheckerTargets` enumeration.

Instantiate the class that describes the diagnostic message and add it to the set of diagnostics:

```csharp
foreach (var name in names)
{
    if (!authorlist.Contains(name))
    {
        // Create the custom error message, including
        // the contextual name information...
        var warning = new AuthorNotCurrentDiagnosticItem(
            this.Context, method.Position, name);

        // and add it to the set of reported messages.
        this.ExtensionContext.AddErrorMessage(warning);
    }
}
```

At this point, you have a complete best practice rule, ready to provide value in your organization. Build it and fix any errors.

**Metadata based Best Practice rules**

Until now we have been describing how to write rules that deal with code. In this section we show how to author rules that apply to metadata, not code. Classes that deal with metadata rules are derived from `BestPracticeMetadataChecker`. The derived instance receives an instance of the metadata describing the artifact that must be checked. You then use the APIs in the `Microsoft.Dynamics.AX.Metadata.Metamodel` to fetch further metadata as needed, and use LINQ queries over the metadata graphs. The template for best practice checks contains a class performing metadata checks as well as a code based one we discussed in the previous section. The mechanics involved in issuing diagnostic messages is the same as we covered above.

**Install, run, and test your rule**

When your code compiles cleanly, a DLL will be created. In order for the tooling to be able to pick up the new rule, this DLL must be installed before running it. Installing the DLL can be done in two ways:
By using the button on the Best Practice configuration dialog. Click the **Install extension** button. You will be asked to point to the assembly file that contains your rule (that is, the DLL generated when you build the rule). Press OK, and the system will copy the DLL where it needs to be (see below).

By manually installing the DLL into the C:\Packages\bin\BPExtensions folder.

If you want to debug your rule, you will find it useful to copy the .pdb file to the same directory as the assembly. After the DLL has been deployed to the target directory, Visual Studio needs to be restarted. After that, the rule is available for use. You may have to debug your rule to iron out any remaining kinks. In fact, stepping through your rule and inspecting the ASTs is valuable when you are getting started. To debug a rule, you need to know that the best practice rule is run by the **xppAgent** process; it is therefore not run within the context of VS itself. Make sure you have selected **Run best practice checks** in the Visual Studio Options dialog, in the **Finance and Operations** page. Otherwise, your check will not run. Set a breakpoint in the **VisitMethod** method, and then do a build of a model that has the new rule switched on as shown above for the Fleet management model. Attach your VS instance to the **xppcAgent** process. When you do a build your breakpoint will be hit, and you can start drilling into your code. You can see all the properties, the list of declarations and statements, and find out all the details about them.

**Running rules in XppBp.exe**

As described above the best practice rules are often run as part of the build of a project from Visual Studio, but there is also a dedicated command-line tool to run them. This tools is the **xppBp.exe** tool, and it is intended mainly for nightly build scenarios. Invoking it from the command line yields a useful overview of the command-line switches and arguments. Here are some useful examples:

- Run BP on all forms in a module: `xppb -module:FleetManagement form:*`
- Run BP on specific elements: `xppb -module:FleetManagement class:MyClass form:MyForm`
- Run BP on all items in the model (and only for this one model in the module): `xppb -module:FleetManagement -model:FleetManagement –all`
- Run BP on all items in all models in the module: `xppb -module:FleetManagement –all`
- Write the output to log files: `xppb -module:FleetManagement -all -xmllog=Log.xml -log=Log.txt`
This topic describes the properties that appear in the Properties window of Microsoft Visual Studio for items in Application Explorer.

Many nodes in Application Explorer represent elements that have properties associated with them. You can read or modify these properties in the Properties window of Microsoft Visual Studio.

### System and common properties

Most application objects in Application Explorer have a standard set of system properties. These system properties are read-only. You can use the Properties window to view the properties for any item in Application Explorer. To open the Properties window, right-click a node in Application Explorer, and then click Properties. On the Categories tab of the Properties window, many system properties are listed under the Statistics node. This article lists additional common properties that are repeated on many, but not all, Application Explorer nodes. The following table shows the system properties that are found on almost all Application Explorer nodes. All these system properties are read-only.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChangedBy</td>
<td>The user who last changed the object (often the release version).</td>
</tr>
<tr>
<td>ChangedDate</td>
<td>The date when the object was last changed.</td>
</tr>
<tr>
<td>ChangedTime</td>
<td>The time when the object was last changed.</td>
</tr>
<tr>
<td>CreatedBy</td>
<td>The user who created the object.</td>
</tr>
<tr>
<td>CreationDate</td>
<td>The date when the object was created.</td>
</tr>
<tr>
<td>CreationTime</td>
<td>The time when the object was created.</td>
</tr>
</tbody>
</table>

The following table shows other common properties that are found on many, but not all, Application Explorer nodes.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConfigurationKey</td>
<td>Specify the configuration key that controls access to or display of an element. If a user doesn't have access to the configuration key, the element isn't visible. Elements include pages, controls on pages, tables, and other elements.</td>
</tr>
<tr>
<td>LegacyID</td>
<td>An identifier element from an earlier version. During upgrade from a previous version, the old identifier is assigned to LegacyID. An installation-specific identifier isn't assigned, and business logic remains intact. This property isn't used for new elements.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NeededAccessLevel</td>
<td>The minimum access level that a user requires. This property is read-only.</td>
</tr>
<tr>
<td>Origin</td>
<td>The globally unique identifier (GUID) of an Application Explorer element. This property is used to identify elements during synchronization and upgrade scenarios. It's a read-only property, and the value never changes after the system assigns it. No origin GUID value is duplicated anywhere in the system.</td>
</tr>
<tr>
<td>SecurityKey</td>
<td>This property is obsolete but is retained for reference in systems that were upgraded from an earlier version.</td>
</tr>
</tbody>
</table>

**Base enum properties**

The following table describes the properties that are available for enumerations.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| AnalysisUsage     | Specify the role of the enumeration in a cube. This setting is automatically propagated to all table fields that reference the enumeration. However, you can override the setting on a table field. The following options are available:  
- **Attribute** – A field that references the enumeration is a dimension attribute.  
- **None** – A field that references the enumeration isn't a dimension attribute. |
| ConfigurationKey  | Specify the configuration key.                                                                                                                                                                            |
| CountryRegionCodes| Specify the codes for the countries/regions where the view is applicable or valid. This property is implemented as a comma-separated list of International Organization for Standardization (ISO) country/region codes in a single string. The values must match data in the global address book. The client framework and application might use this property to enable or disable country/region-specific features. |
| DisplayLength     | Specify the number of characters that are shown. The default value is **Auto**.                                                                                                                              |
| Help              | Create a Help string for the field. The Help string is shown when the field is used on a page.                                                                                                                  |
| Label             | Specify the label that is shown on pages and reports.                                                                                                                                                     |
| Model             | Specify the model that the table is in. A model is a logical grouping of elements in a layer. Examples of elements include a table and a class. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in a higher layer. |
Name

Specify the enumeration name. An enumeration name must indicate either the possible enumeration values or the type of the enumeration value. Examples of enumerations that are named according to the possible values are **InclExcl** and **NextPrevious**. Examples of enumerations that are named according to the type of the enumeration value are **ArrivalPostingType** and **ListStatus**.

Style

Change the default appearance of the enumeration. The following options are available:
- Combo box
- Radio button

UseEnumValue

A value of **Yes** indicates that default values of the **EnumValue** property were modified. A value of **No** resets the **EnumValue** property to the default values.

Extended data type properties

Extended data type (EDT) properties are divided into the following groups, based on whether they are common to all EDTs or available only for certain base data types.

**Properties that are common to all EDTs**

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment</td>
<td>Change the alignment of the text. The available options are Left, Right, and Center.</td>
</tr>
<tr>
<td>AnalysisDefaultSort</td>
<td>Specify the default sort order for a field in a report model that has this EDT.</td>
</tr>
</tbody>
</table>
| AnalysisDefaultTotal    | Specify the aggregate function for a measure. Use this property when the **AnalysisUsage** property is set to **Measure**. The following options are available:  
  - **Sum** – Return the sum of all the values in a set.  
  - **Count** – Return the number of non-null items in a set.  
  - **CountDistinct** – Return the number of distinct non-null items in a set.  
  - **Min** – Return the minimum value in a set.  
  - **Max** – Return the maximum value in a set.  
  - **None** – No aggregate function is applied.  
  - **Auto** – This option applies to derived EDTs. The value of the **AnalysisUsage** property for the parent EDT is used.  
  You can override the aggregate function at the field level. In other words, you can change the aggregate function for the field by using the **AnalysisDefaultTotal** property for that field. |
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnalysisGrouping</td>
<td>Specify whether a field that has this EDT is grouped by default when the field is added to a report by using Report Builder for Microsoft SQL Server Reporting Services (SSRS). This property is automatically set to Discouraged for currency amounts. For other fields that are unique, you should set this property to Discouraged.</td>
</tr>
</tbody>
</table>
| AnalysisUsage          | Specify the role of the EDT in a cube. This setting is automatically propagated to all table fields that reference the EDT. However, you can override the setting on a table field. The following options are available:  
  - **Attribute** – A field that references the EDT is a dimension attribute.  
  - **Measure** – A field that references the EDT is a measure.  
  - **Both** – A field that references the EDT is both a dimension attribute and a measure.  
  - **None** – A field that references the EDT is neither a dimension attribute nor a measure.  
  - **Auto** – This option applies to derived EDTs. The value of the AnalysisUsage property for the parent EDT is used.  
  
  **Note:** Data types that are based on enumerations can't be measures. |
| ArrayLength            | This property is a read-only. The default value is 1. To add array elements to the EDT, right-click the Array Element node, and then click New Array Element. The value of the ArrayLength property is increased to reflect this change. |
| ButtonImage            | Specify the image that is shown when the EDT is used for a lookup button on a page. The following options are available:  
  - **Arrow**  
  - **Mail** – You can select this option for the e-mail type, for example.  
  - **URL** – You can select this option for the URL type, for example.  
  - **ThreeDots** (...)  
  - **OpenFile** – You can select this option for the FilenameOpen and FilenameSave types, for example.  
  - **Calendar** – You can select this option for date types, for example.  
  
  The default value is Arrow. |
<p>| CollectionLabel        | Specify the label that is used to show the plural name of a field that has this EDT. |
| ConfigurationKey      | Specify the configuration key for the EDT. |</p>
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CountryRegionCodes</td>
<td>Specify the codes for the countries/regions where the menu is applicable or valid. This property is implemented as a comma-separated list of ISO country codes in a single string. The values must match data in the global address book. The client uses this property to enable or disable country/region-specific features.</td>
</tr>
<tr>
<td>DisplayLength</td>
<td>Specify the maximum number of characters that are shown on a page or report.</td>
</tr>
<tr>
<td>EnumType</td>
<td>Specify an enumerated data type. This property must be set for EDTs of the enum type.</td>
</tr>
<tr>
<td>Extends</td>
<td>Use this property to base the EDT on another EDT.</td>
</tr>
<tr>
<td>FormHelp</td>
<td>Specify the page to use when you perform a lookup from a field on a page.</td>
</tr>
<tr>
<td>HelpText</td>
<td>Create a Help string for the EDT. The Help string is shown when the type is used on a page.</td>
</tr>
<tr>
<td>ID</td>
<td>This property is read-only.</td>
</tr>
<tr>
<td>Label</td>
<td>Specify the label that is used for the type when the type is used on a page or report.</td>
</tr>
<tr>
<td>Model</td>
<td>Specify the model that the table is in. A model is a logical grouping of elements in a layer. Examples of elements include a table and a class. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in a higher layer.</td>
</tr>
<tr>
<td>Name</td>
<td>Specify the name of the type. The name is used to refer to the type from X++.</td>
</tr>
<tr>
<td>PresenceClass</td>
<td>Specify the X++ class that is used together with the PresenceMethod property to return an instance of the PresenceInfo object.</td>
</tr>
<tr>
<td>PresenceIndicatorAllowed</td>
<td>Specify whether the control that references the EDT should use presence. The default value is Yes.</td>
</tr>
<tr>
<td>PresenceMethod</td>
<td>For the X++ class that is specified in the PresenceClass property, specify the X++ static class method that should be called by using a controls data value. This method returns an instance of the PresenceInfo object that contains the data that the Presence indicator requires.</td>
</tr>
<tr>
<td>ReferenceTable</td>
<td>Specify the table that is referenced by this EDT, and that has the primary key. In other words, this property indicates the primary key table that this EDT references.</td>
</tr>
</tbody>
</table>
Change the default appearance of the EDT. The following options are available:

- Auto
- Combo box
- Radio button

### Properties that are available for only some base data types

Unless the following table specifies otherwise, you should leave all these properties set to Auto.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>TYPE THAT THE PROPERTY EXISTS FOR</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjustment</td>
<td>String</td>
<td>For strings of fixed length, specify whether the characters that are entered should be stored on the left side or the right side of the padding spaces. The available options are Left and Right. The default value is Left.</td>
</tr>
<tr>
<td>AllowNegative</td>
<td>IntegerInt64Real</td>
<td>Specify whether the field can accept negative values.</td>
</tr>
<tr>
<td>AutoInsSeparator</td>
<td>Real</td>
<td>Specify whether the system should automatically insert a decimal separator. For example, if you enter 2222, the system automatically shows 2222.00.</td>
</tr>
<tr>
<td>ChangeCase</td>
<td>String</td>
<td>Specify how text that is entered in a string control should be formatted. For example, the text can be formatted as all uppercase letters, or it can use title capitalization. Note: This property isn't supported for Enterprise Portal.</td>
</tr>
<tr>
<td>DateDay</td>
<td>DateUtcDateTime</td>
<td>Specify how the day should be shown.</td>
</tr>
<tr>
<td>DateFormat</td>
<td>DateUtcDateTime</td>
<td>Specify the layout of a date.</td>
</tr>
<tr>
<td>DateMonth</td>
<td>DateUtcDateTime</td>
<td>Specify how the month should be shown.</td>
</tr>
<tr>
<td>DateSeparator</td>
<td>DateUtcDateTime</td>
<td>Specify the separators between year, month, and day.</td>
</tr>
<tr>
<td>DateYear</td>
<td>DateUtcDateTime</td>
<td>Specify how the year should be shown.</td>
</tr>
<tr>
<td>DecimalSeparator</td>
<td>Real</td>
<td>Specify the decimal separator. When the default setting (Auto) is used, the decimal separator that is specified in the system setup is used.</td>
</tr>
<tr>
<td>DisplaceNegative</td>
<td>IntegerInt64Real</td>
<td>Specify whether to align negative numbers to the left.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>TYPE THAT THE PROPERTY EXISTS FOR</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DisplayHeight</td>
<td>String</td>
<td>Specify the number of lines to show at the same time when the EDT is shown on a page.</td>
</tr>
<tr>
<td>EnumType</td>
<td>Enum</td>
<td>Specify the base enum that is used to create the EDT.</td>
</tr>
</tbody>
</table>
| FormatMST        | Real                              | Specify master currency values should be formatted. The following options are available:  
|                  |                                   | • Auto  
|                  |                                   | • Yes  
|                  |                                   | • No  
|                  |                                   | The default value is **Auto**.                                               |
| NoOfDecimals     | Real                              | Specify the number of decimal places when a value is shown on a page or a report. |
| RotateSign       | IntegerInt64Real                  | Select this option to reverse the sign for the number. In other words, change a minus sign (–) to a plus sign (+) or a plus sign to a minus sign. |
| ShowZero         | IntegerInt64Real                  | Specify whether to show a field that has a value of 0 (zero) as an empty field. If a value of 0 in fields of this type means null/nothing, set this property to **No**. |
| SignDisplay      | IntegerInt64Real                  | Specify whether to show the sign of a negative number, and also whether the sign should appear before or after the number. Typically, you should set this property to **Auto**. However, you can set it to **None** if the **DisplaceNegative** property is used. |
| StringSize       | String                            | Specify the maximum size of the string.                                     |
| ThousandSeparator| Real                              | Specify the symbol that is used to separate thousands.                      |
| TimeFormat       | TimeUtcDateTime                   | Specify how times should be formatted.                                     |
| TimeHours        | TimeUtcDateTime                   | Specify whether to include hours.                                          |
| TimeMinute       | TimeUtcDateTime                   | Specify whether to include minutes.                                        |
| TimeSeconds      | TimeUtcDateTime                   | Specify whether to include seconds.                                        |
| TimeSeparator    | TimeUtcDateTime                   | Specify the separator that is used for times.                              |
In Application Explorer, under the **Data Dictionary** node, there is a **Perspectives** node. A perspective is a collection of tables and views that contain the measures and dimensions for a cube. The following table describes the properties that can be set for each perspective. For information about the system properties that are available for a perspective, see the “System and common properties” section. For information about the properties for tables that are associated with a perspective, see the “Table properties” and “Table field properties” sections.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConfigurationKey</td>
<td>Specify the configuration key that is assigned to the perspective. The configuration key determines which configurations of a perspective are included for in report models that are generated.</td>
</tr>
<tr>
<td>HelpText</td>
<td>Create a string to use as a description for the perspective in a report model.</td>
</tr>
<tr>
<td>ID</td>
<td>Specify the identifier of the perspective.</td>
</tr>
<tr>
<td>Label</td>
<td>Specify the name that is shown for the perspective in a report model.</td>
</tr>
<tr>
<td>Model</td>
<td>Specify the model that the perspective is in. A model is a logical grouping of elements in a layer. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in a higher layer.</td>
</tr>
<tr>
<td>SharedDimensionContainer</td>
<td>Specify whether to share items in the perspective. When this property is set to <strong>Yes</strong>, the items in the perspective are added to all other perspectives that are in the project, and no cube is created for the perspective. The default value is <strong>No</strong>.</td>
</tr>
</tbody>
</table>
| Usage                  | Specify the materialization options for a perspective. The following options are available:  
  - **AdHocReporting** – The perspective will be used to generate a transactional Semantic Model Definition Language (SMDL) model.  
  - **OLAP** – The perspective will be used to generate a cube in a Microsoft SQL Server Analysis Services (SSAS) Business Intelligence project.  
  - **Both** – The perspective will be used to generate both a transactional SDML model and a cube in an SSAS Business Intelligence project.  
  - **None** – The perspective won’t be materialized.  
  The default value is **None**. |
Table properties

This section describes the properties that appear in the Properties window for table elements in Application Explorer. Table elements are located under Data Dictionary > Tables.

**Table properties**

The following table describes the properties of table elements in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>Specify whether the table supports inheritance. The default value is No. If the value is set to Yes, the table can't be a direct target of X++ SQL statements such as update_recordset and select. <strong>Note:</strong> This property is unavailable when the SupportInheritance property is set to No.</td>
</tr>
</tbody>
</table>
| AnalysisDimensionType | Specify the type of dimension that is created, based on the setting of the IsLookup property. If the IsLookup property is set to Yes, the following options are available:  
  - **Auto** – The table can contain both factual and dimensional data. The BI Wizard will extract dimensional data, and create dimensions and attributes. Factual data will be extracted to create measures. One child dimension is created that has attributes from the parent table.  
  - **MasterInner** – An inner (full) join is used to create relationships with this table to the child table. Each record combination for this table and the child table are generated in the dimension. One child dimension is created that has attributes from the parent table.  
  - **MasterLeftOuter** – A left outer join is used to create relationships with this table to the child table. Dimensions will have additional attributes, based on values in this table that can also be empty. One child dimension is created that has attributes from the parent table.  
  - **Transaction** – The table should be used to generate only factual data (measures). You should use this option when a table contains only transactional data. One child dimension is created that contains only enumeration fields from the table.  

If the IsLookup property is set to No, the following options are available:  
  - **Auto** – Table can contain both factual and dimensional data. The BI Wizard will extract dimensional data, and create dimensions and attributes. Factual data will be extracted to create measures. One parent and child dimension are created.  
  - **MasterInner** – Not applicable. This option is the same as Auto.  
  - **MasterLeftOuter** – Not applicable. This option is the same as Auto.  
  - **Transaction** – The table should be used to generate only factual data (measures). You should use this option when a table contains only transactional data. One child dimension is created that contains only enumeration values from the table.
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnalysisIdentifier</td>
<td>Specify the field to use as the identifier for the dimension in an SSAS cube.</td>
</tr>
<tr>
<td>AOSAuthorization</td>
<td>Specify the type of operation that a user can perform on a table, depending on the user's permissions. When this property is set to <strong>None</strong>, no authorization check is performed.</td>
</tr>
<tr>
<td>CacheLookup</td>
<td>Specify how to cache the records that are retrieved during a lookup operation. This property exists only on tables that don't inherit from another table. On an inheritance root table, you can't set this property to <strong>EntireTable</strong> by using the Application Explorer Properties window. You must not use other techniques to assign this value to inheritance root tables. For example, don't use the AOTsetProperty method of the TreeNode class to assign this value.</td>
</tr>
<tr>
<td>ClusterIndex</td>
<td>Specify the cluster index. This property is used only for SQL optimization.</td>
</tr>
<tr>
<td>ConfigurationKey</td>
<td>Specify the configuration key for the table. Configuration keys let a system administrator enable and disable specific parts of an application.</td>
</tr>
<tr>
<td>CountryRegionCodes</td>
<td>Specify the codes for the countries/regions where the table is applicable or valid. This property is implemented as a comma-separated list of ISO country codes in a single string. The values must match data in the global address book. The client framework uses this property to enable or disable country/region-specific features.</td>
</tr>
<tr>
<td>CountryRegionContextField</td>
<td>Specify the field that is used to identify the country/region context. This property is related to the CountryRegionCodes property.</td>
</tr>
<tr>
<td>CreatedBy</td>
<td>Specify whether the system maintains the CreatedBy field for the records in a table. This field contains information about the person who created a record.</td>
</tr>
<tr>
<td>CreatedDateTime</td>
<td>Specify whether the system maintains the CreationDate and CreationTime fields for the records in a table. This field contains the date when a record was created.</td>
</tr>
<tr>
<td>CreatedTransactionId</td>
<td>Specify whether the system maintains the CreatedTransactionId field for the records in a table. This field contains information about the transaction that created a record.</td>
</tr>
<tr>
<td>CreateRecIdIndex</td>
<td>Specify whether an index on the Record ID field is created.</td>
</tr>
<tr>
<td>DeveloperDocumentation</td>
<td>Describe the purpose of a table, and explain how it's used in the program. Typically, a description contains no more than five sentences and is written as a single paragraph.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EntityRelationshipType</td>
<td>Classify a table according to common entity relationship (ER) data model notation. A table is classified as either an entity or a relationship. An entity represents an object, whereas a relationship represents an association between two objects.</td>
</tr>
<tr>
<td>Extends</td>
<td>Derive the table from the specified table. The value of this property is null when the SupportInheritance property is set to Yes.</td>
</tr>
<tr>
<td>FormRef</td>
<td>Specify the display menu item that is activated when a table is referenced. A display menu item is associated with a page. When you use a primary index field on a report, this page is available as a link in the report. You specify a primary index by using the PrimaryIndex property. If you leave this property blank, the system tries to display a page that has the same name as the table.</td>
</tr>
<tr>
<td>ID</td>
<td>The system-generated table ID.</td>
</tr>
</tbody>
</table>
| IsLookup                     | For report models, use this property to specify whether the table information is incorporated into other tables that reference it when a report model is generated. For online analytical processing (OLAP) cubes, use this property to specify whether to generate a consolidated dimension or a distinct dimension. The following options are available:  
  - **Yes** – Attributes from the table should be consolidated into the parent dimension (star schema).  
  - **No** – A separate dimension should be generated for the table (snowflake schema). |
<p>| Label                        | Specify the label for a table.                                                                                                                                                                             |
| ListPageRef                  | Specify a display menu item that points to a page that can show a list of this record type.                                                                                                                                 |
| Model                        | Specify the model that the table is in. A model is a logical grouping of elements in a layer. Examples of elements include a table and a class. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in a higher layer. |
| ModifiedBy                   | Specify whether the system maintains the ModifiedBy field for the records in a table. This field contains information about the person who last modified a record.                                               |
| ModifiedDateTime             | Specify whether the system maintains the ModifiedDateTime field for the records in a table. This field contains the date when a record was last modified.                                                      |
| ModifiedTime                 | Specify whether the system maintains the ModifiedDateTime field for the records in a table. This field contains the date and time when a record was last modified.                                              |
| Name                         | Specify the table name.                                                                                                                                                                                    |</p>
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>OccEnabled</td>
<td>Specify whether the optimistic concurrency mode is enabled for a table. When this mode is enabled, data isn’t locked from future modification when it’s fetched from the database. Data is locked only when the actual update is performed.</td>
</tr>
<tr>
<td>PreviewPartRef</td>
<td>Specify the info part or form part to use in the enhanced preview. An info part shows a collection of data fields from a specified query. It uses metadata to describe how the data appears. A form part represents a pointer to a page.</td>
</tr>
<tr>
<td>PrimaryIndex</td>
<td>Specify the primary index. Only a unique index can be selected. This property is used for database optimization and to indicate which unique index should be used as the caching key. If you don’t specify a primary index, the unique index that has the lowest ID is used as the caching key.</td>
</tr>
<tr>
<td>ReplacementKey</td>
<td>Specify the fields to display as the identifier for data in some page controls.</td>
</tr>
<tr>
<td>ReportRef</td>
<td>Specify the output menu item that is activated when a table is referenced. An output menu item is associated with a report. When you use a primary index field on a report, this report is available as a link in the report. You specify a primary index by using the PrimaryIndex property.</td>
</tr>
<tr>
<td>SaveDataPerCompany</td>
<td>Specify whether the data for the current company is saved. If you set the property to No, data is saved without a company identifier (DataAreaId). <strong>Note:</strong> If the SaveDataPerCompany property on a table is set to Yes, the SetCompany property on a page design that uses the table as a data source must also be set to Yes. <strong>Tip:</strong> The status line shows the acronym for the company. Double-click the acronym to open a dialog box where you can change the company.</td>
</tr>
<tr>
<td>SaveDataPerPartition</td>
<td>A value that indicates whether the table has a system field that is named Partition. This property is intended to be read-only. If the table has a Partition field, each record is assigned to one partition. Each record is hidden from data access operations that are run under the context of other partitions.</td>
</tr>
<tr>
<td>SearchLinkRefName</td>
<td>Specify the name of the menu item that links to information on a website about a table record that is listed in the Enterprise Portal search results. If the SearchLinkRefType property is set to URL, select a menu item that links to a Web Part page that shows the table data. Forms and reports on Web Part pages can display data.</td>
</tr>
<tr>
<td>SearchLinkRefType</td>
<td>Specify the type of the menu item that links to information on a website about a table record that is listed in the Enterprise Portal search results.</td>
</tr>
<tr>
<td>SingularLabel</td>
<td>Specify the label that is used in a report model or a cube to show the singular name of items that are stored in the table.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SupportInheritance</td>
<td>When you set this property to <strong>Yes</strong>, you can set a value for other inheritance-related properties, such as <strong>Extends</strong> and <strong>Abstract</strong>. <strong>Caution:</strong> If you set this property to <strong>Yes</strong>, any fields on the table are dropped and must be created again.</td>
</tr>
<tr>
<td>SystemTable</td>
<td>Indicate whether a table appears as a system table. A table that appears as a system table can be filtered during export and import. System tables are always synchronized when you sign in. Therefore, this property might be useful for tables that you use as soon as you sign in.</td>
</tr>
</tbody>
</table>
| TableContents             | Specify how setup/parameter data can be reused from one customer to another. The following options are available:  
  - **Not specified** – Use this option for most tables.  
  - **Default Data** – Use this option for customer-independent data, such as postal codes, units, and time intervals.  
  - **Base Data** – Use this option for customer-dependent data, such as calendars, groups, and parameters.  
  - **Default+Base data** – Use this option for data where the local perception varies. For example, Chart of Accounts is customer-independent in Germany but is customer dependent in most other places. |
| TableGroup                | Specify the group that the table belongs to. Table groups provide a method for categorizing tables according to the type of data that they contain. You can use table groups to define whether the system should prompt users when they update or delete data from the table on pages by using the table as the data source. When you export data, you can use table groups to filter records. |
| TableType                 | This property replaces the **Temporary** property that is found in Microsoft Dynamics AX 2009.                                                                 |
| TitleField1, TitleField2  | You can use this property in the following ways:  
  - Add table field data to a form caption.  
  - Show additional fields on a lookup page. The **TitleField1** property is also used when you activate the lookup list in a field on a page. The fields that you specify for the **TitleField1** and **TitleField2** properties can be merged with the key value.  
  - Show field information in a tooltip. |
<p>| TypicalRowCount           | Specify the number of records that typically appear in a table. If the <strong>AnalysisSelection</strong> property isn't set, this property determines how records are selected by using Report Builder for SSRS. The setting of this property affects whether a drop-down list, a list box, or a filtered list box is used to select table records. |
| ValidTimeStateFieldType   | Specify the type of date-time field that the system uses when it tracks data within time spans.                                                |</p>
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visible</td>
<td>Specify the access rights when the table is used as a data source on a page or a report. If the table is used as a data source on a page, the access rights on the page can’t exceed the access rights that are defined for the table.</td>
</tr>
</tbody>
</table>

### Tables and report models

The following properties are related to report models that are used to add information to a report:

- AnalysisSelection
- AnalysisVisibility
- IsLookup
- SingularLabel
- TypicalRowCount

### Table field properties

The following properties are related to report models that are used to add information to a report:

- AnalysisDefaultTotal
- AnalysisLabel
- AnalysisTotaling
- AnalysisUsage
- AnalysisVisibility
- CurrencyCode
- CurrencyCodeField
- CurrencyCodeTable

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjustment</td>
<td>Specify whether the string field should be left-aligned or right-aligned when it's stored in the database. For example, if the 11-character string &quot;hello world&quot; is stored in a right-aligned field that has a <code>StringSize</code> setting of 40, 29 space characters are stored as the prefix. <strong>Note:</strong> The <code>Adjustment</code> setting affects the search results when you search for a value in a table by using the &gt;, &lt;, &gt;=, and &lt;= relational operators. It doesn't affect the search results when you use the == operator. The <code>Adjustment</code> setting is ignored when the <code>StringSize</code> property is set to (Memo).</td>
</tr>
<tr>
<td>AliasFor</td>
<td>Specify the table field that the field is an alias for.</td>
</tr>
<tr>
<td>AllowEdit</td>
<td>Specify whether users are allowed to modify data in an existing record on a page.</td>
</tr>
<tr>
<td>AllowEditOnCreate</td>
<td>Specify whether users are allowed to enter data in the field when a new record is created from a page.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| AnalysisDefaultTotal        | For report models, use this property to specify how field data is aggregated when an automatic total for the table is displayed in a report that is built by using SSRS and report models. The default value is **No**, which indicates that the field isn't automatically shown as a total. For OLAP cubes, use this property to specify the aggregate function for a measure. Use this property when the **AnalysisUsage** property is set to **Measure**. The following options are available:  
  - **Sum** – Return the sum of all the values in a set.  
  - **Count** – Return the number of non-null items in a set.  
  - **CountDistinct** – Return the number of distinct non-null items in a set.  
  - **Min** – Return the minimum value in a set.  
  - **Max** – Return the maximum value in a set.  
  - **None** – No aggregate function is applied.  
  - **Auto** – This option applies to derived EDTs. The value of the **AnalysisUsage** property for the parent EDT is used.                                                                                                                                                                                                                                           |
| AnalysisLabel               | Specify the label to use as the caption in an SSAS cube for the table field. The label is applied to either a dimension attribute or a measure. This property is intended for situations where one of the following conditions is true:  
  - The **Label** property isn't defined.  
  - The **Label** property doesn't work as a caption for a dimension attribute or a measure in a SSAS cube.                                                                                                                                                                                                                                                               |
| AnalysisUsage               | Specify the role of the field in a cube. The following options are available  
  - **Attribute** – The field is a dimension attribute.  
  - **Measure** – The field is a measure.  
  - **Both** – The field is both a dimension attribute and a measure.  
  - **None** – The field is neither a dimension attribute nor a measure.  
  - **Auto** – The value of the **AnalysisUsage** property for the EDT or enumeration that the field is based on should be used.                                                                                                                                                                                                                                                                                                 |
<p>| ConfigurationKey            | Set the configuration key for the field.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| CountryRegionCodes          | Specify the codes for the countries/regions where the table field is applicable or valid. This property is implemented as a comma-separated list of ISO country codes in a single string. The values must match data in the global address book. The client framework and application might use this property to enable or disable country/region-specific features.                                                                                                                                                                                                                       |
| CountryRegionContextField   | Specify the field that is used to identify the country/region context. See the description of the <strong>CountryRegionCodes</strong> property.                                                                                                                                                                                                                                                                                                                                                             |</p>
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExtendedDataType</td>
<td>Specify the EDT to use for this field.</td>
</tr>
<tr>
<td>GroupPrompt</td>
<td>Specify a label that is used for the field when it appears in a group. Tip: You can use this property to help guarantee that a field label doesn’t repeat text that appears in the label for a field group. For example, if a field group on a page is labeled Customer, don’t include this text in the GroupPrompt property for fields that are included in the field group.</td>
</tr>
<tr>
<td>HelpText</td>
<td>Specify the Help string for the field. The Help string is shown when the field is used on a page.</td>
</tr>
<tr>
<td>ID</td>
<td>The system-generated field ID.</td>
</tr>
<tr>
<td>IgnoreEDTRelation</td>
<td>This property is used during the migration of EDT relations. When you migrate relations from an EDT node to a table node, you can skip an invalid relation for a given table field. To skip invalid relations, set this property to Yes. The default value is No.</td>
</tr>
<tr>
<td>Label</td>
<td>Specify a label for the field. This label will appear on pages and reports. Also see the description of the AnalysisLabel property earlier in this table.</td>
</tr>
</tbody>
</table>
| Mandatory           | Specify whether a user must add data to a field on a page. Set this property to Yes to indicate that the default or initialization value for each data type isn’t acceptable for persistence into the database. The following list shows some default values that can’t be used for mandatory fields on a page:  
  • Empty isn’t acceptable for a str (string) field.  
  • The minimum date-time isn’t acceptable for date-time fields such as date and utcdatetime.  
  • A value of 0 (zero) isn’t acceptable for numeric fields such as int, real, and enum.  
  Finance and Operations doesn’t support the semantics for the null value that is standard in most SQL database products. Field can’t be null in the database. Therefore, the Mandatory property has nothing to do with the concept of a null value. Caution: A mandatory table field can have its EnumType property set to an enumeration. You might define a field as an enum type that includes an item that has the integer value 0. In this case, 0 isn’t an item that’s available for selection on the page. The forms system automatically calls the validateWrite method to enforce the setting of the Mandatory property. However, the Mandatory property has no effect on the behavior of direct X++ SQL that inserts or updates the value of a table field. In your direct X++ SQL, you can include calls to the validateWrite method on your table buffer variable. Your buffer variable inherits the method from the xRecord class. |
MinReadAccess

Specify the mode of the automatic authorization feature. Automatic authorization has two modes of operation: surrogate foreign key and lookup. If a table in a query is tagged for surrogate foreign key authorization, and the user doesn't have access to that table but hasn't been explicitly denied, view access is granted to the table. However, not all fields will be visible. The visibility is determined by the following rules:
- If MinReadAccess is set to No, no access is granted to the field.
- If MinReadAccess is set to Yes, view access is granted to the field.
- Otherwise, view access is granted if the field is part of the natural key automatic identification group, if it's a title field, or if it's a system field.

If a table in a query is tagged for lookup authorization, access is determined by the following rules:
- If MinReadAccess is set to No, no access is granted to the field.
- Otherwise, view access is granted to the field.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| MinReadAccess     | Specify the mode of the automatic authorization feature. Automatic authorization has two modes of operation: surrogate foreign key and lookup. If a table in a query is tagged for surrogate foreign key authorization, and the user doesn't have access to that table but hasn't been explicitly denied, view access is granted to the table. However, not all fields will be visible. The visibility is determined by the following rules:
- If MinReadAccess is set to No, no access is granted to the field.
- If MinReadAccess is set to Yes, view access is granted to the field.
- Otherwise, view access is granted if the field is part of the natural key automatic identification group, if it's a title field, or if it's a system field. |
| Model             | Specify the model that the table field is in. A model is a logical grouping of elements in a layer. Examples of elements include a table or class. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in a higher layer. |
| Name              | Specify the name of the field. |
| RelationContext   | Specify the mapping of a field to a specific table relation. Typically, this property is used in unit-of-measure scenarios to model data that is related to currency codes or quantities. The relation that is associated with the field can then be used to show a lookup of currency codes or quantities. There is no default value. |
| SaveContents      | Specify whether the field data is saved in the database or treated as virtual field data. Virtual field data is calculated at run time when the field is displayed. This data has no physical representation in the database. Tip: Instead of virtual fields, you can use display and edit methods. |
| StringSize        | Set the field length, in the number of characters. The maximum field length depends on the database. A value of (Memo) indicates that the field length is unlimited. |
| Type              | Specify the base type of a field. |
| Visible           | Specify whether the field should be visible in the user interface. |

Table index properties

The following table describes the properties that are available for indexes on tables.
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowDuplicates</td>
<td>If you set this property to Yes, the index can be non-unique. If you don't create at least one unique index, a unique index is created by combining the first index and the RecId.</td>
</tr>
<tr>
<td>AlternateKey</td>
<td>Specify whether this index is part of an alternate key. The index field must have a unique value in every record.</td>
</tr>
<tr>
<td>ConfigurationKey</td>
<td>Set the configuration key. An index field that is disabled through a configuration key is automatically removed from the index.</td>
</tr>
<tr>
<td>Enabled</td>
<td>You can use this property to disable the index.</td>
</tr>
<tr>
<td>ID</td>
<td>The internal identifier of the object.</td>
</tr>
<tr>
<td>Model</td>
<td>Specify the model that the table index is in. A model is a logical grouping of elements in a layer. Examples of elements include a table or class. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in a higher layer.</td>
</tr>
<tr>
<td>Name</td>
<td>Specify the index name.</td>
</tr>
<tr>
<td>UniqueAcrossCompanies</td>
<td>This property is for internal Microsoft use only. The available values are Yes and No. The default value is No. The value of this property is ignored when the AllowDuplicates property is set to No. However, when AllowDuplicates is set to Yes, a value of Yes for UniqueAcrossCompanies can improve the performance of some cross-company queries. The performance improvement is caused by changes to the caching of data.</td>
</tr>
<tr>
<td>ValidTimeStateKey</td>
<td>Specify whether this index key is used to determine the valid time state relationship with the parent table. The default value is No. Tip: To enable this property, you must set the AllowDuplicates property to No and the AlternateKey property to Yes.</td>
</tr>
<tr>
<td>ValidTimeStateMode</td>
<td>Specify whether gaps are allowed between two date-effective records. The default value is NoGap. Tip: To enable this property, you must set the AllowDuplicates property to No, the AlternateKey property to Yes, and the ValidTimeStateKey property to Yes.</td>
</tr>
</tbody>
</table>

**NOTE**
Pages sort on the first index.

**Table relation properties**

**List of properties**
The following table describes the properties for a table relation in Application Explorer.
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardinality</td>
<td>The number of times that each primary key value from the referenced table must occur in the foreign key column of the current table. For example, the <strong>OneMore</strong> value means one or more, but not zero. This value indicates that every parent key value must occur in the child table's foreign key column at least one time. A relation node under a SalesLine table might use the <strong>OneMore</strong> value when the business rule requires that every record in the parent SalesTable table relate to at least one item that is being sold. Currently, the <strong>Cardinality</strong> property is not used. However, future releases might use this property and the <strong>RelatedTableCardinality</strong> property.</td>
</tr>
<tr>
<td>CreateNavigationPropertyMethods</td>
<td>A value of <strong>Yes</strong> instructs the system to generate navigation methods on the table buffer class for each foreign key relation node.</td>
</tr>
<tr>
<td>EDTRelation</td>
<td>If the value is set to <strong>Yes</strong>, a software tool was used to migrate this relation to its current location from an old EDT relation.</td>
</tr>
<tr>
<td>EntityRelationshipRole</td>
<td>This property is used to clarify the semantics of a relationship that is defined on a table. A role name should be either a noun or a noun phrase. The role name should indicate the role of the associated table in relation to the associating object. Alternatively, the role name should be a short phrase that starts with a present-tense verb that indicates the role that the table plays in the relationship. Role names aren't required when the relationship is unambiguous.</td>
</tr>
<tr>
<td>Model</td>
<td>The model that this relation is part of.</td>
</tr>
<tr>
<td>Name</td>
<td>A descriptive name that you choose for the relation.</td>
</tr>
<tr>
<td>NavigationPropertyMethodNameOverride</td>
<td>Specify the name of the navigation method. If no value is specified, the navigation method uses the value from the <strong>RelatedTableRole</strong> property.</td>
</tr>
<tr>
<td>RelatedTableCardinality</td>
<td>Specify whether the foreign key field value in the current table can be <strong>null</strong> in some or all records of the current table. The following options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>ZeroOne</strong> means zero or one. This value indicates that the foreign key field in a child record can be <strong>null</strong>.</td>
</tr>
<tr>
<td></td>
<td>• <strong>ExactlyOne</strong> indicates that the foreign key field can't be <strong>null</strong> in any child record.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RelatedTableRole</td>
<td>Enter a text value to describe the purpose of the referenced parent table in this relationship. When a table has only one relation that references a given parent table, you can use the name of the parent table. Sometimes, a table has more than one relation to a given referenced parent table. In this case, the value of the RelatedTableRole property should describe the relation well enough to distinguish the relation's purpose from the other relation to the same parent table. The value of this property can be used as the value of the JoinRelation property of a data source relation under an Application Explorer query. In standard cases, this usage is recommended, because it reduces denormalization. This property interacts with the UseDefaultRoleNames property.</td>
</tr>
<tr>
<td>RelationshipType</td>
<td>Select a value that describes the subtle relationship between two tables. For example, the Composition value indicates that the child record can't meaningfully exist unless it's related to a specific parent record. The record for the fourth floor in the Floor table can't exist unless it references a record in the parent Building table. <strong>Note:</strong> The DeleteActions should be compatible with this property setting. For a Composition relationship, the DeleteActions should include delete cascade behavior. Currently, the RelationshipType property is not used. However, a future release might use this property.</td>
</tr>
<tr>
<td>Role</td>
<td>Specify a name that describes the meaning or role of the relation. For example, one relation to a Department table could track the department that the employee currently belongs to. Another relation could track the department that the employee has requested a transfer to. Although both these relations are relations to the Department table, they fill different roles. As the value of this property, it's a good idea to join the names of the child table and parent table by using an underscore (_) character. For example, enter <strong>SalesTable_SalesLine</strong>. This property interacts with the UseDefaultRoleNames property.</td>
</tr>
<tr>
<td>Table</td>
<td>The table that the relation refers to.</td>
</tr>
<tr>
<td>UseDefaultRoleNames</td>
<td>A value of <strong>Yes</strong> indicates that the system must generate default values for the Role and RelatedTableRole properties. Even when this property is set to <strong>Yes</strong>, the values for that are generated for Role and RelatedTableRole don't appear in the Properties window. Additionally, the TreeNode class doesn't use the generate values. However, the DictRelation reflection class does use the generated values.</td>
</tr>
</tbody>
</table>
A value of **Yes** indicates that when a page inserts a record into the child table, the insertion is rejected unless the related record exists in the referenced parent table. Additionally, when a page deletes a record from the parent table, the deletion is either rejected or cascades to the related records in the child table. Set the value to **No** when the **RelationshipType** property is set to **Link**. You might also set the value to **No** in special temporary cases, such as during some upgrade scenarios. When the value is set back to **Yes**, no validation occurs for records that were inserted or deleted while the value was **No**. **Caution**: A value of **Yes** for the **Validate** property doesn’t prevent direct X++ SQL data operations from deleting parent records or inserting child records that violate the integrity of foreign key data.

### RelatedTableRole and query JoinRelation

This section describes how you can use the **RelatedTableRole** property to simplify the creation of a new query. If you enter an explicit value for the **RelatedTableRole** property on a table relation, you can use that value to populate the **JoinRelation** property on a data source relation under a **Queries > MyQuery** node in Application Explorer. You can use this method to specify the fields of the join in one location. If the join fields ever change, you must update the join in only one location. Before you can set a value for the **JoinRelation** property, you must delete the values of the **Field** and **RelatedField** properties.

### CreateNavigationPropertyMethods and RelatedTableRole

When you set the **CreateNavigationPropertyMethods** property to **Yes** on a table relation, the system generates navigation methods for the table buffer class. A navigation method links two table buffer instances by using their foreign key relationship. The **UnitOfWork** class is one area where this navigation linkage is used. The name of a navigation method is copied from the value of the **RelatedTableRole** property on the table relation. This behavior is used when the **RelatedTableRole** value is explicitly set in the **Properties** window, and when the system generates the **RelatedTableRole** value because the **UseDefaultRoleNames** property is set to **Yes**. The property values generate the following navigation method on the child CustTable buffer. Most directly, the navigation method name is copied from the value of the **RelatedTableRole** property.

```java
public final CustBankAccount BankAccounts([CustBankAccount relatedTable])
```

### NavigationPropertyMethodNameOverride property

The following list describes cases where you must override the name that the system generates for a navigation method on a table buffer class:

- The table class already has a method name that matches the values of the **RelatedTableRole** property.
- The value of the **RelatedTableRole** property exceeds the maximum length that can be used for a method name.

In these cases, you must choose a valid name for the navigation method and assign that name as the value of the **NavigationPropertyMethodNameOverride** property on the table relation.

### Understanding the RelationshipType enumeration
When you add a node under table relations, you can set the value of the `RelationshipType` property for the new relation. The list of possible values for the `RelationshipType` property is the list of elements in the `RelationshipType` enumeration. This section describes the meaning of each element in the `RelationshipType` enumeration.

Description of elements

The following table describes the elements of the `RelationshipType` property.

<table>
<thead>
<tr>
<th>ELEMENT NAME</th>
<th>DESCRIPTION</th>
<th>AUTOMATIC INFERENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSpecified</td>
<td>This element is often the default value of the <code>RelationshipType</code> property.</td>
<td>When the <code>RelationshipType</code> property has a value of <code>NotSpecified</code>, the system infers an appropriate value. The system infers the value in the following sequence: 1. Specialization 2. Link 3. Composition 4. Aggregation 5. Association. For example, if the criteria for both Composition and Aggregation are met, the system infers Composition, because Composition occurs earlier in the list.</td>
</tr>
<tr>
<td>Specialization</td>
<td>This element applies only to table inheritance, to relationships between base and derived tables.</td>
<td>The system sets the <code>RelationshipType</code> property to <code>Specialization</code> whenever table inheritance is involved.</td>
</tr>
<tr>
<td>Link</td>
<td>This element is a non-relational relationship. It requires that the <code>Validate</code> property be set to <code>No</code>. This type of relationship supports navigation between pages that list many records from a table and pages that provide detail fields for one record from the table.</td>
<td>Link is used only to support the migration of EDT link relations during upgrade from earlier versions. Migration tools create this type relationship, but you must not.</td>
</tr>
<tr>
<td>Composition</td>
<td>This element is a stronger type of Aggregation relation. A table must not have more than one Composition relation. For example, a building is composed of rooms, and a given room can't exist in more than one building.</td>
<td>If the criteria for Composition are met, but you manually assign a value of Aggregation or Association, the system leaves the value as Aggregation or Association.</td>
</tr>
</tbody>
</table>
Aggregation

This element is appropriate when the child table is considered subordinate to the entity of the parent table. The system infers Aggregation when one of the following conditions is true:

- The parent table has a delete action node that is defined to use this relation node.
- Any foreign key field for this relation in the child table has the Mandatory property set to Yes.

If the criteria for Aggregation are met, but you manually assign a value of Association, the system leaves the value as Association.

Association

This element is the concept of a standard foreign key. You must set the RelationshipType property to Association if the system doesn't set the value of the property to anything, and if both Aggregation and Composition are inappropriate.

View properties

The properties for views are the same as the properties for tables. However, because views are read-only, most of their properties can’t be changed. Some of these properties have fixed values, and some are inherited from the data sources that are used in the query that defines the view. The following properties for views are related to data analysis when you’re using SSRS. All these properties can be changed.

- AnalysisVisibility
- AnalysisSelection
- TypicalRowCount
- IsLookup
- SingularLabel

The following table describes the properties that can be set for a view.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AOSAuthorization</td>
<td>Use this property to specify which data access operations require verification of user permissions.</td>
</tr>
<tr>
<td>CacheLookup</td>
<td>The record cache level for the table.</td>
</tr>
<tr>
<td>ClusterIndex</td>
<td>The cluster index of the table, if there is a cluster index.</td>
</tr>
<tr>
<td>ConfigurationKey</td>
<td>Set the configuration key for the view.</td>
</tr>
<tr>
<td>CountryRegionCodes</td>
<td>Specify the codes for the countries/regions where the menu is applicable or valid. This property is implemented as a comma-separated list of ISO country codes in a single string. The values must match data in the global address book. The client uses this property to enable or disable country/region-specific features.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CountryRegionContextField</td>
<td>Specify the field that is used to identify the country/region context. See</td>
</tr>
<tr>
<td></td>
<td>the description of the <code>CountryRegionCodes</code> property.</td>
</tr>
<tr>
<td>DeveloperDocumentation</td>
<td>Describe the purpose of a view, and explain how it's used in the program.</td>
</tr>
<tr>
<td></td>
<td>Typically, a description contains no more than five sentences and is written</td>
</tr>
<tr>
<td></td>
<td>as a single paragraph.</td>
</tr>
<tr>
<td>EntityRelationshipType</td>
<td>Classify a view according to common entity relationship (ER) data model</td>
</tr>
<tr>
<td></td>
<td>notation. A view is classified as either an entity or a relationship. An</td>
</tr>
<tr>
<td></td>
<td>entity represents an object, whereas a relationship represents an association</td>
</tr>
<tr>
<td></td>
<td>between two objects.</td>
</tr>
<tr>
<td>FormRef</td>
<td>Specify the default page for the view. The default page is the page that</td>
</tr>
<tr>
<td></td>
<td>is shown when the user activates Jump to Main Table by using the shortcut</td>
</tr>
<tr>
<td></td>
<td>menu for a field on a page. The page is referenced through a menu item of</td>
</tr>
<tr>
<td></td>
<td>the <code>Display</code> type. If you leave this property blank, MorphX tries to</td>
</tr>
<tr>
<td></td>
<td>activate a page that has the same name as the table that you're referring</td>
</tr>
<tr>
<td></td>
<td>to.</td>
</tr>
<tr>
<td>ID</td>
<td>The internal identifier of the object.</td>
</tr>
<tr>
<td>Label</td>
<td>Specify a user-friendly name for the view.</td>
</tr>
<tr>
<td>ListPageRef</td>
<td>Specify a display menu item that points to a page that can show a list of</td>
</tr>
<tr>
<td></td>
<td>this record type.</td>
</tr>
<tr>
<td>Model</td>
<td>Specify the model that the view is in. A model is a logical grouping of</td>
</tr>
<tr>
<td></td>
<td>elements in a layer. An element can exist in exactly one model in a layer.</td>
</tr>
<tr>
<td></td>
<td>The same element can exist in a customized version in a model that is in a</td>
</tr>
<tr>
<td></td>
<td>higher layer.</td>
</tr>
<tr>
<td>Name</td>
<td>Specify the name of the view. This name is used when you refer to the view</td>
</tr>
<tr>
<td></td>
<td>from the X++ language.</td>
</tr>
<tr>
<td>PreviewPartRef</td>
<td>Specify the info part or form part to use in the enhanced preview. An info</td>
</tr>
<tr>
<td></td>
<td>part shows a collection of data fields from a specified query. It uses</td>
</tr>
<tr>
<td></td>
<td>metadata to describe how the data appears. A form part represents a pointer</td>
</tr>
<tr>
<td></td>
<td>to a page.</td>
</tr>
<tr>
<td>PrimaryIndex</td>
<td>Specify the primary index of the view. Only a unique index can be selected.</td>
</tr>
<tr>
<td></td>
<td>This property is used for database optimization and to indicate which unique</td>
</tr>
<tr>
<td></td>
<td>index should be used as the caching key. If you don't specify a primary</td>
</tr>
<tr>
<td></td>
<td>index, the unique index that has the lowest ID is used as the caching key.</td>
</tr>
<tr>
<td>Query</td>
<td>Specify the query that is the source of data for the view. You can use this</td>
</tr>
<tr>
<td></td>
<td>property instead of adding data sources directly to the view.</td>
</tr>
<tr>
<td>ReportRef</td>
<td>The name of the default report for the table.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SaveDataPerCompany</td>
<td>Set this property to <strong>Yes</strong> for company-specific tables. Set it to <strong>No</strong> if the data is related to cross-companies, installation, a database, Application Explorer, tracing, or OLAP. For example, the SysTraceTable or OLAPServerTable table specifies whether data should be saved for that table on a per-company basis, or whether the data should be available without any company affiliation. If the <strong>SaveDataPerCompany</strong> property on a table is set to <strong>Yes</strong>, that table has a <strong>DataAreaId</strong> column that contains the company identifier. If the table property is set to <strong>No</strong>, the <strong>DataAreaId</strong> column is removed from the table.</td>
</tr>
<tr>
<td>SaveDataPerPartition</td>
<td>A value that indicates whether the view has a system field that is named <strong>Partition</strong>. This property is intended to be read-only. If the view has a <strong>Partition</strong> field, each record is assigned to one partition. Each record is hidden from data access operations that are run under the context of other partitions.</td>
</tr>
<tr>
<td>SearchLinkRefName</td>
<td>The name of the menu item that links to information on a website about a table record that is listed in the Enterprise Portal search results.</td>
</tr>
<tr>
<td>SearchLinkRefType</td>
<td>The type of the menu item that links to information on a website about a table record that is listed in the Enterprise Portal search results.</td>
</tr>
<tr>
<td>SystemTable</td>
<td>A value that indicates whether a table is a system table. System tables can be filtered during export and import, and are always synchronized when you sign in. Therefore, this property might be useful for tables that you use as soon as you sign in.</td>
</tr>
</tbody>
</table>
| TableContents          | Specify how setup/parameter data can be reused from one customer to another. The following options are available:  
  • **Not specified**  – Use this option for most tables.  
  • **Default Data**  – Use this option for customer-independent data, such as postal codes, units, and time intervals.  
  • **Base Data**  – Use this option for customer-dependent data, such as calendars, groups, and parameters.  
  • **Default+Base data**  – Use this option for data where the local perception varies. For example, Chart of Accounts is customer-independent in Germany but is customer dependent in most other places. |
### Property Description

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>TitleField1, TitleField2</td>
<td>The information that is shown in the window caption for the view. The caption is constructed from the following elements:</td>
</tr>
<tr>
<td></td>
<td>• The TitleField1 label, followed by colon (:) and a space</td>
</tr>
<tr>
<td></td>
<td>• The value of the current record in the column that is used for TitleField1, followed by a comma (,)</td>
</tr>
<tr>
<td></td>
<td>• The value of the current record in the column that is used for TitleField2</td>
</tr>
<tr>
<td>ValidTimeStateEnabled</td>
<td>Specify whether the view supports the valid time state feature of the underlying table. The default value is No. You can set this property to Yes only if both the following conditions are true:</td>
</tr>
<tr>
<td></td>
<td>• The underlying table is a valid time state table.</td>
</tr>
<tr>
<td></td>
<td>• The view has ValidFrom and ValidTo in its Fields list.</td>
</tr>
<tr>
<td>Visible</td>
<td>Specify the access rights when the table is used as a data source on a page or a report. If the table is used as a data source on a page, the access rights on the page can't exceed the access rights that are defined for the table.</td>
</tr>
</tbody>
</table>

### Data set properties

This section contains descriptions of the properties on data set elements in Application Explorer. The Data Sets node is a high-level node in Application Explorer. Data sets are used to access data in Enterprise Portal.

**Description of properties**

The following table describes the properties that are available on data set nodes in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Set the name of the data set.</td>
</tr>
</tbody>
</table>

**Data Sources properties**

The following table describes the properties for the Data Sources node of the data set.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChangeGroupMode</td>
<td>Specify how changes to the data sources are committed. The following options are available:</td>
</tr>
<tr>
<td></td>
<td>• None – The changes to any data source for the data set are committed independently of changes to the other data sources.</td>
</tr>
<tr>
<td></td>
<td>• ImplicitInnerOuter – All the data sources that are inner-joined or outer-joined work as a single unit. All changes are committed successfully, or they are rolled back if an error occurs.</td>
</tr>
</tbody>
</table>

### Data set data source properties

The following table describes the properties that are available for data set data sources.
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowCheck</td>
<td>Specify whether security checks occur before the data set is accessed. The following options are available:</td>
</tr>
<tr>
<td></td>
<td>- <strong>Yes</strong> – The user's read permissions are verified before the data set is accessed.</td>
</tr>
<tr>
<td></td>
<td>- <strong>No</strong> – The user's read permissions are verified only after the data set is accessed. No data is retrieved if the user lacks sufficient permission for the underlying data sources.</td>
</tr>
<tr>
<td></td>
<td><em>Yes</em> is the default value and is usually recommended.</td>
</tr>
<tr>
<td>AllowCreate</td>
<td>Specify whether users can create new records in the data source (that is, in the table for the data source).</td>
</tr>
<tr>
<td>AllowDelete</td>
<td>Specify whether users can delete records in the data source (that is, in the table for the data source).</td>
</tr>
<tr>
<td>AllowEdit</td>
<td>Specify whether users can modify the data. <strong>Tip:</strong> You can set the <strong>AllowEdit</strong> property for the whole data source here. The same property also exists on each field in the data source, so that you can prohibit modifications for individual fields.</td>
</tr>
<tr>
<td>AutoNotify</td>
<td>This property isn't used for data sets.</td>
</tr>
<tr>
<td>AutoQuery</td>
<td>This property isn't used for data sets.</td>
</tr>
<tr>
<td>AutoSearch</td>
<td>This property isn't used for data sets.</td>
</tr>
<tr>
<td>CounterField</td>
<td>Specify one of the fields in the data source as a counter for the data set. The field must be an index on the underlying table for the data source, and it must be of the <strong>real</strong> type. This property helps guarantee that a record that is inserted in a data set has a line number that corresponds to the actual sequential position in the data. For example, if a new line is inserted between lines 3 and 4, the new line becomes line number 3.5.</td>
</tr>
<tr>
<td>CrossCompanyAutoQuery</td>
<td>Specify whether the data source retrieves data from more than one company database.</td>
</tr>
<tr>
<td>DelayActive</td>
<td><strong>Use this property to delay execution of the active method for the data source.</strong> If you set this property to <strong>Yes</strong>, the active method is activated only after a delay of 20 milliseconds. When a user scrolls through a data source, the active method isn't called on every record. Instead, it's called only on the final record that the user selects. <strong>Tip:</strong> The <strong>DelayActive</strong> property is particularly useful when two data sources are linked (that is, when the <strong>LinkType</strong> property is set to <strong>Delayed</strong>). This property is part of the AutoJoin system.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Index</td>
<td>Set the index that is used to specify a sorting order. You can choose any of the indexes on the table. If you specify an index in this manner, it's used as an index hint on each query to the database. The index specifies both an access path and a sort order for the records in the data set, based on this data source. The initial sort order for the records is prioritized in this manner: 1. If sort fields are added to the data source query, the sort specification is used. 2. If an index is specified in the Index property on the data source, the sort order that is implicitly specified in that index is used. 3. If the data source is autojoined with another data source, the system finds the most appropriate index for this join and then sorts the data according to that index. 4. If nothing else is specified, the sort order that is implicitly specified in the first index (the index that has the lowest ID) on the table that is used in the page data source is used. When no index hints are specified, the database management system locates an applicable access path. This access path is based on the information in the query that is supplied. The user can change the sort order for a page by using the query dialog box.</td>
</tr>
<tr>
<td>InsertAtEnd</td>
<td>Specify whether a new record is created when the user moves focus past the last record in the table.</td>
</tr>
<tr>
<td>InsertIfEmpty</td>
<td>Specify whether a blank record is inserted if there are no records in the table. If you set this property to No, you must manually create a new record.</td>
</tr>
<tr>
<td>JoinSource</td>
<td>Use this property to join two data sources. Set this property when two or more tables are used as the data source, and you want to join them.</td>
</tr>
<tr>
<td>LinkType</td>
<td>Use this property to maintain an active link between two data sources. When focus changes in the first data source, the corresponding record or records in the second data source are selected. For example, a customer table and a table of transactions is used for each customer. When the user scroll from one customer to the next, the transaction list is automatically updated to show transactions for the current customer. Set this property to Delayed for the outer (externally linked) data source. The linked data source is updated only after a delay of 100 milliseconds. This delay helps guarantee that the linked data source isn't updated while the user is scrolling through a data source. The update occurs only after the user finally focuses on a record. This property is part of the AutoJoin system.</td>
</tr>
<tr>
<td>Name</td>
<td>Set the name of the data source. This name should be the same as the name of the underlying table.</td>
</tr>
</tbody>
</table>
**OnlyFetchActive**
Specify whether to fetch all fields in the data source or only those fields that are used by the data set. When this property is set to **Yes**, records can't be deleted from the data set. This restriction helps preserve data integrity, because it helps guarantee that a delete operation is never tried on incomplete records.

**OptionalRecordMode**
Specify the create and delete behavior for records on an outer-joined table. The following options are available:
- **ImplicitCreate** – When no record is saved in the database, create an outer-joined record and joined tables as soon as the parent record becomes active. If the outer-joined record or its children aren't changed, they will be deleted when the parent record is no longer active.
- **ExplicitCreate** – When no record is saved in the database, treat this record as disabled until the user explicitly triggers creation by using the **Optional Record** check box. When the record exists, clearing the check box will delete this record.
- **None** – No special create or delete behavior occurs for an outer-joined record.

**StartPosition**
Specify whether the first record or the last record should be the current record when the data set is accessed.

**Table**
Set the table that is used as the data source.

**ValidTimeStateAutoQuery**
Specify the types of queries for date effectivity (**AsOfDate** or **DateRange**).

**ValidTimeStateUpdate**
Specify the types of updates for an existing date-effective record. The following options are available:
- **CreateNewTimePeriod** – On the record that is becoming the previous record, the **ValidTo** date field is set to a date that is no later than the current date. In the same transaction, the new current record has its **ValidFrom** field set to immediately after **ValidTo** date of the previous record.
- **Correction** – The **ValidFrom** or **ValidTo** value of existing rows must be modified to keep the date-effective data valid after the record set is updated.
- **EffectiveBased** – Records in the past can't be edited. Records that are currently active are edited in a manner that resembles CreateNewTimePeriod mode. Future records are edited in a manner that resembles Correction mode.

The default value is **CreateNewTimePeriod**.

---

**Form properties**

This section describes the properties that you set on forms in Application Explorer. To provide a uniform application interface, many properties have **Auto** values. You can create forms by using a drag-and-drop operation and then manually setting several properties. To specify the name of a form, you set the **Name** property in the **Properties** window for the form. All other properties on the top-level node for the form are
Form design properties

Most properties on the **Design** node for a form also exist on the individual controls. Examples include the **Width** and **Height** properties. However, when you set a property on the **Design** node instead of setting it on a control, the setting affects the whole form. A few properties exist only on the **Design** node. The following table describes these properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlignChild</td>
<td>Specify whether a control within a group follows the <strong>AlignChildren</strong> property setting for the group or for the overall form design. For example, <strong>AlignChildren</strong> is set to <strong>Yes</strong> on the <strong>Design</strong> node for the form, but you don't want a particular group to be arranged together with the other groups. In this case, set <strong>AlignChild</strong> to <strong>No</strong> for that group.</td>
</tr>
<tr>
<td>AlignChildren</td>
<td>Align the child controls within a container.</td>
</tr>
<tr>
<td>AllowDocking</td>
<td>Specify whether a form can be attached to the client workspace. The default value is <strong>No</strong>.</td>
</tr>
<tr>
<td>AllowFormCompanyChange</td>
<td>Specify whether the form supports company changes when it's used as a child form with a cross-company dynamic-link library (DLL). The default value is <strong>No</strong>.</td>
</tr>
</tbody>
</table>
| AllowUserSetUp                | Specify whether a user can move controls on a form and can change the value of control properties. This property is also found on the design of a form. The following options are available:  
  • **No** – Users can't customize any controls in this container.  
  • **Restricted** – Users can change properties of individual controls, but they can't move controls.  
  • **Yes** – There are no restrictions on the user setup.  
The default value is **Yes**. **Caution**: Full user setup isn't allowed if any of the parent containers for the control have restrictions on the user setup level. The **AllowAdd** property on form data sources determines whether a user can add a field to a form. |
| AlwaysOnTop                   | Specify whether the form always appears on top of other windows in the z-order. The default value is **No**.                                  |
| ArrangeMethod                 | Specify whether to arrange child field groups in columns or in rows.                                                                       |
| ArrangeWhen                   | Specify when the controls in the container should be arranged. The following options are available:  
  • **Startup**  
  • **On demand**  
  • **Never**  
  • **Default**  
  • **Auto**  
The default value is **Startup**. |
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BackgroundColor</strong></td>
<td>Specify the color that is used for the background of the control. To make the background opaque or transparent, use the <strong>BackStyle</strong> property.</td>
</tr>
<tr>
<td><strong>BottomMargin</strong></td>
<td>Set the bottom margin of the form in pixels. The default value is <strong>Auto</strong>.</td>
</tr>
<tr>
<td><strong>Caption</strong></td>
<td>Specify the heading for grouped controls. Use a label for this property.</td>
</tr>
<tr>
<td><strong>ColorScheme</strong></td>
<td>Specify the color palette for the control. To change the color palette for the whole form, set the <strong>ColorScheme</strong> property for the largest container, and keep the default values for the individual controls.</td>
</tr>
<tr>
<td><strong>Columns</strong></td>
<td>Specify the number of columns that show the information. <strong>Caution:</strong> Field groups on the underlying table are never split into more than one column.</td>
</tr>
<tr>
<td><strong>ColumnSpace</strong></td>
<td>Set the amount of space between columns in container controls.</td>
</tr>
<tr>
<td><strong>DataSource</strong></td>
<td>Specify the table that data in the control comes from. To set a particular field within the table, use the <strong>DataField</strong> property. If the control opens another form, relations between the data source for the control, as specified by this property, and the data source on the other form help guarantee that records in the second form are dynamically selected. For example, a customer is selected in one form, and the control opens a form that shows customer transactions. In this case, the second form shows a range of customer transactions that apply to the current customer. <strong>Caution:</strong> If you set the <strong>DataSource</strong> and <strong>DataField</strong> properties, their settings override any settings for the <strong>DataMethod</strong> or <strong>ExtendedDataType</strong> properties.</td>
</tr>
<tr>
<td><strong>Font</strong></td>
<td>Change the font properties for the control by using the <strong>Font</strong> dialog box. Use the dialog font to specify the font, font style, and font size.</td>
</tr>
<tr>
<td><strong>Frame</strong></td>
<td>Specify the frame style that the form uses.</td>
</tr>
<tr>
<td><strong>Height</strong></td>
<td>Specify the height of the form or control in pixels.</td>
</tr>
<tr>
<td><strong>HideIfEmpty</strong></td>
<td>Use this property to hide a container control if it’s empty. This property has no affect if the <strong>Width</strong> and <strong>Height</strong> properties of the container are set to <strong>Auto</strong>, because the size of the control is 0 (zero) in this case.</td>
</tr>
<tr>
<td><strong>HideToolBar</strong></td>
<td>Hide form-specific buttons on the toolbar.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ImageMode</td>
<td>Define how the bitmap that is specified by the ImageName property appears in a control. The following options are available:</td>
</tr>
<tr>
<td></td>
<td>- Normal</td>
</tr>
<tr>
<td></td>
<td>- Size to fit</td>
</tr>
<tr>
<td></td>
<td>- Side by side</td>
</tr>
<tr>
<td></td>
<td>- Center</td>
</tr>
<tr>
<td></td>
<td>The default value is Normal.</td>
</tr>
<tr>
<td>ImageName</td>
<td>Specify the image that is shown for a control. You can select only .bmp files. To use one of the resource files, use the ImageResource property instead.</td>
</tr>
<tr>
<td>ImageResource</td>
<td>Use one of the images from the image resource file as the image for a control. Specify the ID of the image. You can select only an image from the integrated resource file. To use another file type, use the ImageName property.</td>
</tr>
<tr>
<td>LabelFont</td>
<td>Change the font for the text that is supplied in the Label property.</td>
</tr>
<tr>
<td>Left</td>
<td>Change the position of the upper-left corner of the form. There are several predefined settings. You can also specify an exact position in pixels. The following predefined settings are available:</td>
</tr>
<tr>
<td></td>
<td>- Auto (left)</td>
</tr>
<tr>
<td></td>
<td>- Auto (right)</td>
</tr>
<tr>
<td></td>
<td>- Left edge</td>
</tr>
<tr>
<td></td>
<td>- Right edge</td>
</tr>
<tr>
<td></td>
<td>- Center</td>
</tr>
<tr>
<td></td>
<td>The default value is Auto (left).</td>
</tr>
<tr>
<td>LeftMargin</td>
<td>Change the default left margin of the form. The margin is specified in pixels.</td>
</tr>
<tr>
<td>MaximizeBox</td>
<td>Specify whether to include the maximize box in the upper-right corner of the enclosing window. The default value is Yes.</td>
</tr>
<tr>
<td>MinimizeBox</td>
<td>Specify whether to include the minimize box in the upper-right corner of the enclosing window. The default value is Yes.</td>
</tr>
<tr>
<td>Mode</td>
<td>Specify the data entry mode for the form.</td>
</tr>
<tr>
<td>Model</td>
<td>Specify the model that the form is in. A model is a logical grouping of elements in a layer. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in a higher layer.</td>
</tr>
<tr>
<td>RightMargin</td>
<td>Change the default right margin of the form. The margin is specified in pixels.</td>
</tr>
<tr>
<td>SaveSize</td>
<td>Set this property to Yes to save the size of the form.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ScrollBars</td>
<td>Specify whether scroll bars are enabled in the form.</td>
</tr>
<tr>
<td>SetCompany</td>
<td>Cause the system to change the company when the form receives focus. <strong>Note:</strong> If the <code>SaveDataPerCompany</code> property on a table is set to <code>Yes</code>, the <code>SetCompany</code> property on a form design that uses the table as a data source must also be set to <code>Yes</code>.</td>
</tr>
<tr>
<td>StatusBarStyle</td>
<td>Specify how the status bar appears in a form. Use this property to hide the status bar, show only Help information, show status bar elements according to the <code>WindowType</code> setting, or always show the full status bar. <strong>Note:</strong> Forms that have a <code>WindowType</code> setting of <code>ListPage</code>, <code>ContentPage</code>, or <code>Workspace</code> ignore this property.</td>
</tr>
<tr>
<td>Style</td>
<td>Specify the style of the form. This property controls the form design pattern that is used for the form. The following options are available:</td>
</tr>
<tr>
<td></td>
<td>• Auto</td>
</tr>
<tr>
<td></td>
<td>• DetailsFormMaster</td>
</tr>
<tr>
<td></td>
<td>• DetailsFormTransaction</td>
</tr>
<tr>
<td></td>
<td>• Dialog</td>
</tr>
<tr>
<td></td>
<td>• DropDialog</td>
</tr>
<tr>
<td></td>
<td>• FormPart</td>
</tr>
<tr>
<td></td>
<td>• ListPage</td>
</tr>
<tr>
<td></td>
<td>• Lookup</td>
</tr>
<tr>
<td></td>
<td>• SimpleList</td>
</tr>
<tr>
<td></td>
<td>• SimpleListDetails</td>
</tr>
<tr>
<td></td>
<td>• TableOfContents</td>
</tr>
<tr>
<td></td>
<td>The default value is <strong>Auto</strong>.</td>
</tr>
<tr>
<td>TitleDataSource</td>
<td>Specify the data source to use in the form caption.</td>
</tr>
<tr>
<td>Top</td>
<td>Change the position of the top of the form. There are several predefined settings. You can also specify an exact position in pixels. The following predefined settings are available:</td>
</tr>
<tr>
<td></td>
<td>• Auto</td>
</tr>
<tr>
<td></td>
<td>• Top edge</td>
</tr>
<tr>
<td></td>
<td>• Bottom edge</td>
</tr>
<tr>
<td></td>
<td>• Center</td>
</tr>
<tr>
<td></td>
<td>The default value is <strong>Auto</strong>.</td>
</tr>
<tr>
<td>TopMargin</td>
<td>Set the top margin of the form in pixels. The default value is <strong>Auto</strong>.</td>
</tr>
<tr>
<td>UseCaptionFromMenuItem</td>
<td>Specify whether to replace the form caption with the label from the calling menu item. This property enables the caption of the form to be changed when the form is opened. The default value is <strong>No</strong>.</td>
</tr>
</tbody>
</table>
**PROPERTY** | **DESCRIPTION**
---|---
ViewEditMode | Specify whether the form opens in read-only mode or as a form that allows you to change fields. The following options are available:  
- **View** – Open the form as read-only.  
- **Edit** – Open the form in edit mode.  
- **Auto** – Open the form in the appropriate mode.  
The default value is Auto.

Visible | Use this property to hide the form. **Caution**: You can't use the Visible property to enforce access restrictions. The user can change the visibility for the controls in the Form Setup dialog box. To enforce access restrictions, use the Enabled and NeededAccessLevel properties instead.

Width | Change the width of the form in pixels.

WindowResize | Specify whether the form can be resized.

WindowType | Specify the type of window.

WorkflowDataSource | Set the root data source for the workflow on a form. The root data source that you specify should be the same root data source that is specified in the query that used for the Document property on the workflow template.

WorkflowEnabled | Set this property to Yes to enable the workflow menu bar on the form. The default value is No.

WorkflowType | Specify the workflow type, which determines the following items and behaviors:  
- The workflow document to use. The workflow document exposes calculated fields and identifies the query that exposes data fields for the workflow.  
- Whether the user can configure fields and approvals.  
- The workflow categories to use when a workflow type is assigned to a specific module.  
- Menu items and event handlers.

---

**Help document set properties**

A document set is a collection of Help documentation that is associated with a workspace. When you publish a content element, you use metadata to add your content element or table of contents information to a document set. To manage the relationship between a workspace and a document set, Application Explorer includes a node that is named **Help Document Sets**. Each document set in the Help Document Sets node includes a collection of properties. You edit these properties when you add a new document set or change the relationship between a document set and a workspace. **Caution**: A workspace can be associated with only one document set. Although Application Explorer lets you add a new document set and associate it with a workspace, you will no longer see documentation from the document set that you replaced. Typically, you use **UserDocumentation** as the document set for any content element or table of contents entries that you publish to the Help server. The following table describes the properties for a document set in the Help Document Sets node of Application Explorer.
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocumentSetName</td>
<td>String</td>
<td>A name that uniquely identifies the document set. The name is limited to 40 characters and must not contain whitespace. Use the value of this property when you set the value of the DocumentSets metadata element in a content element or table of contents file.</td>
</tr>
<tr>
<td>DocumentSetDescription</td>
<td>String</td>
<td>The text or label to display for the document set. This value appears in the Search content from list of the Options menu of the Help viewer.</td>
</tr>
<tr>
<td>AddToApplicationHelpMenu</td>
<td>Boolean</td>
<td>Set this property to Yes if you want the document set to appear on the Help menu of the application workspace.</td>
</tr>
<tr>
<td>AddToDeveloperHelpMenu</td>
<td>Boolean</td>
<td>Set this property to Yes if you want the document set to appear on the Help menu of the developer workspace.</td>
</tr>
<tr>
<td>UserDocumentSet</td>
<td>Boolean</td>
<td>Set this property to Yes to associate the document set with the application workspace. If you set this property to No, you can't view the context-sensitive (F1) Help that was published by Microsoft.</td>
</tr>
<tr>
<td>DeveloperDocumentSet</td>
<td>Boolean</td>
<td>Set this property to Yes to associate the document set with the development workspace. If you set this property to No, you can't view the context-sensitive (F1) Help that was published by Microsoft.</td>
</tr>
<tr>
<td>ContentLocation</td>
<td>Enumeration</td>
<td>An enumeration value that specifies where to retrieve documentation. The following table describes the enumeration.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VALUE</th>
<th>LABEL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>

*Note: The table above is not filled in.*
Menu properties

The following table describes the properties that are available for menus under the **Menus** node in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConfigurationKey</td>
<td>Set the configuration key for the menu.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CountryRegionCodes</td>
<td>Specify the codes for the countries/regions where the menu is applicable or valid. This property is implemented as a comma-separated list of ISO country codes in a single string. The values must match data in the global address book. The client uses this property to enable or disable country/region-specific features.</td>
</tr>
<tr>
<td>DisabledImage</td>
<td>Specify the button image that is used when the menu is disabled. If this property isn't set, the system uses the setting of the NormalImage property to generate an image.</td>
</tr>
<tr>
<td>DisabledImageLocation</td>
<td>Specify the location of the image that is used for a disabled control. You can use images from a file, the Resources node in Application Explorer, or an embedded resource. The value that you select for this property determines the values that are available for the DisabledImage property. If the property isn't set, the system uses the setting of the ImageLocation property to generate an image.</td>
</tr>
<tr>
<td>ImageLocation</td>
<td>Specify the location of the image that is used. You can use images from a file, the Resources node in Application Explorer, or an embedded resource. The value that you select for this property determines the values that are available for the NormalImage property.</td>
</tr>
<tr>
<td>Label</td>
<td>Set the name of the menu that is shown to the user.</td>
</tr>
<tr>
<td>MenuItemName</td>
<td>Specify the menu item to include on the menu. The values that are available depend on the value of the MenuItemType property.</td>
</tr>
<tr>
<td>MenuItemType</td>
<td>Specify the type of the menu item. There are three categories of menu items: Display, Output, Action. The value that you set for this property determines the list of menu item names that appears in the list for the MenuItemName property.</td>
</tr>
<tr>
<td>Model</td>
<td>Specify the model that the menu is in. A model is a logical grouping of elements in a layer. Examples of elements include a table or class. An element can be located in exactly one model in a layer. The same element can be located in a customized version in a model that is in a higher layer.</td>
</tr>
<tr>
<td>NormalImage</td>
<td>Specify the image that is used when the menu is enabled.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Specify one or more values that are passed to an object. These values resemble the parameters that are passed to a method. A parameter supplies a value that is then used to perform the task. There is no default value.</td>
</tr>
</tbody>
</table>
**SetCompany**

If you set this property to *Yes*, every time that the menu is opened, the company is changed to the company that was specified when the menu was first started.

**Shortcut**

Specify the keyboard shortcut that opens the menu. For example, you can press Ctrl+F3 to open the menu. There is no default value.

**ShowParentModule**

Specify whether to update the navigation pane, based on the parent module of the menu item. The following options are available:
- *Yes* – Always update the navigation pane, based on the parent module of the menu item.
- *No* – Leave the navigation pane unchanged, even if the parent module of the menu item differs from the current module.

The default value is *Yes*.

### Menu item properties

The following properties are available for all menu items (display, output, and action), even menu items that are used for web menus.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ConfigurationKey</strong></td>
<td>Select the configuration key that is required in order to enable the menu item. Use the key for the module that the object belongs to.</td>
</tr>
<tr>
<td><strong>CopyCallerQuery</strong></td>
<td>Specify whether to copy the query from the calling form to the target form. This property enables the target form to show the same data that was viewed in the original form. The default value is <em>Auto</em>.</td>
</tr>
</tbody>
</table>
| **CorrectPermissions**    | Specify whether correct permission should be available for selection when privileges are assigned to the menu item. The following options are available:  
  - *Auto* – The permission will be available for selection as a privilege on this menu item’s Privileges node under the Entry Points node.  
  - *No* – The permission won’t be available for selection as a privilege on the menu item.  

The default value is *Auto*. |
<p>| <strong>CountryConfigurationKey</strong> | Optional: Set a country/region-specific key in addition to or instead of a standard configuration key. |
| <strong>CountryRegionCodes</strong>    | Specify the codes for the countries/regions where the menu item is valid. This property is implemented as a comma-separated list of ISO country codes in a single string. The values must match data in the global address book. The client uses this property to enable or disable country/region-specific features. |</p>
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| CreatePermissions            | Specify whether create permission should be available for selection when privileges are assigned to the menu item. The following options are available:  
• **Auto** – The permission will be available for selection as a privilege on this menu item’s Privileges node under the Entry Points node.  
• **No** – The permission won’t be available for selection as a privilege on the menu item.  
The default value is **Auto**.                                                                 |
| DeletePermissions            | Specify whether delete permission should be available for selection when privileges are assigned to the menu item. The following options are available:  
• **Auto** – The permission will be available for selection as a privilege on this menu item’s Privileges node under the Entry Points node.  
• **No** – The permission won’t be available for selection as a privilege on the menu item.  
The default value is **Auto**.                                                                 |
| DisabledImage                | Specify the image that is used when the menu item is disabled. If this property isn’t set, the system uses the setting of the NormalImage property to generate an image. |
| DisabledImageLocation        | Specify the location of the image that is used for a disabled control. You can use images from a file, the Resources node in Application Explorer, or an embedded resource. The value that you select for this property determines the values that are available for the DisabledImage property. If the property isn’t set, the system uses the setting of the ImageLocation property to generate an image. |
| EnumTypeParameter and EnumParameter | Optional: Select an enumerated type as a parameter for the object, and then select an enumeration value as the value of the EnumParameter property. Typically, these properties are used when one form is used in several situations. You can change the behavior of the form, depending on the EnumParameter value. For example, the PriceDiscGroup form is used by three display menu items (PriceDiscGroup_*), each of which has a different EnumParameter value. In the form’s init method, a switch construct validates the value, and then the form is created. |
| ExtendedDataSecurity         | Specify whether the menu item appears under all companies instead of in the context of a single company. The default value is **No**.                 |
| FormViewOption               | Specify the form mode to use. The following options are available:  
• **Auto**  
• **Grid**  
• **Details**  
The default value is **Auto**.                                                                 |

**PROPERTY** and **DESCRIPTION**

---

**CreatePermissions**  
Specify whether create permission should be available for selection when privileges are assigned to the menu item. The following options are available:  
• **Auto** – The permission will be available for selection as a privilege on this menu item’s Privileges node under the Entry Points node.  
• **No** – The permission won’t be available for selection as a privilege on the menu item.  
The default value is **Auto**.

**DeletePermissions**  
Specify whether delete permission should be available for selection when privileges are assigned to the menu item. The following options are available:  
• **Auto** – The permission will be available for selection as a privilege on this menu item’s Privileges node under the Entry Points node.  
• **No** – The permission won’t be available for selection as a privilege on the menu item.  
The default value is **Auto**.

**DisabledImage**  
Specify the image that is used when the menu item is disabled. If this property isn’t set, the system uses the setting of the NormalImage property to generate an image.

**DisabledImageLocation**  
Specify the location of the image that is used for a disabled control. You can use images from a file, the Resources node in Application Explorer, or an embedded resource. The value that you select for this property determines the values that are available for the DisabledImage property. If the property isn’t set, the system uses the setting of the ImageLocation property to generate an image.

**EnumTypeParameter and EnumParameter**  
Optional: Select an enumerated type as a parameter for the object, and then select an enumeration value as the value of the EnumParameter property. Typically, these properties are used when one form is used in several situations. You can change the behavior of the form, depending on the EnumParameter value. For example, the PriceDiscGroup form is used by three display menu items (PriceDiscGroup_*), each of which has a different EnumParameter value. In the form’s init method, a switch construct validates the value, and then the form is created.

**ExtendedDataSecurity**  
Specify whether the menu item appears under all companies instead of in the context of a single company. The default value is **No**.

**FormViewOption**  
Specify the form mode to use. The following options are available:  
• **Auto**  
• **Grid**  
• **Details**  
The default value is **Auto**.
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelpText</td>
<td>Create a Help string for the menu item. The text appears on the status bar when you select the object that is opened by the menu item (for example, a form). <strong>Note:</strong> To write a Help topic for the menu item, in Application Explorer, in the Application Documentation/Menu Items node, find the topic that has the same name as your menu item. This topic will then be shown instead of any Help topic that was written about the object that is opened by the menu item.</td>
</tr>
<tr>
<td>ImageLocation</td>
<td>Specify the location of the image that is used for a control. You can use images from a file, the Resources node in Application Explorer, or an embedded resource. The value that you select for this property determines the values that are available for the NormalImage property.</td>
</tr>
<tr>
<td>Label</td>
<td>Select the label to use as the name that appears for the item on menus and buttons.</td>
</tr>
<tr>
<td>LinkedPermissionObject</td>
<td>If the permissions of another object (for example, a form or report) should be applied to this menu item, select the object. Typically, this property is used for action menu items.</td>
</tr>
<tr>
<td>LinkedPermissionType</td>
<td>Specify the type of the object that is specified by the LinkedPermissionObject property.</td>
</tr>
<tr>
<td>MultiSelect</td>
<td>Select whether the menu item can be used on multiple record selections in forms.</td>
</tr>
<tr>
<td>Model</td>
<td>Specify the model that the table is in. A model is a logical grouping of elements in a layer. Examples of elements include a table or class. An element can be located in exactly one model in a layer. The same element can be located in a customized version in a model that is in a higher layer.</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the menu item.</td>
</tr>
<tr>
<td>NeededAccessLevel</td>
<td>Define the minimum access that is required for the menu item to appear on a menu or a button. This property is used to set access to the menu item for different user groups.</td>
</tr>
<tr>
<td>NeedsRecord</td>
<td>Specify whether a button that represents the menu item is enabled if no record is present. The default value is No. You can use this property to help guarantee that an action can be completed. For example, you have a menu item button that opens a detail form. You might want to disable the button if there are no records on the list page.</td>
</tr>
<tr>
<td>NormalImage</td>
<td>Specify the image that is used when the menu item is associated with an enabled button control.</td>
</tr>
<tr>
<td>Object</td>
<td>Select an object of the object type that is specified in the Class property.</td>
</tr>
</tbody>
</table>
### PROPERTY | DESCRIPTION
--- | ---
**ObjectType** | Select the type of object that the menu item opens. **Caution:** You should use `SSRSReport` for a menu item for an SSRS report. Don't use `SQLReportLibraryReport` for new menu items. The `SQLReportLibraryReport` option is obsolete and will be removed in a future version.

**OpenMode** | Specify the view mode of the target form. You use this property to specify whether the target form opens in edit mode or read-only mode. The following options are available:
- **Auto**
- **View**
- **Edit**
- **New**
The default value is **Auto**.

**Parameters** | Optional: Specify the arguments that are passed to the object.

**Query** | Select the query that is passed to the target form for the `InitialQuery` method.

**ReadPermissions** | Specify whether read permission should be available for section when privileges are assigned to the menu item. The following options are available:
- **Auto** – The permission will be available for selection as a privilege on this menu item’s **Privileges** node under the **Entry Points** node.
- **No** – The permission won’t be available for selection as a privilege on the menu item.
The default value is **Auto**.

**ReportDesign** | Select the report design to use for a specific SSRS report model.

**RunOn** | Select whether to run the menu item on the client, the server, or the location that it’s called from. This property is mainly used for menu items that open reports. This property determines where the application object is run from only if the `RunOn` property of the object is set to **Called from**.
- A form is instantiated and run on the client, because the **FormRun** class always runs on the client.
- A report is instantiated and run as specified by the menu item’s **RunOn** property, because the **ReportRun** class always runs where it was called from. You should set the property to **Called from**. If you set the report to run on the client, and the report is run in a batch, the report will fail. If you set the report to run on the server, and the report is shown on the screen, the report will fail.
- A class’s **main** method is run as specified by its modifier. The class itself is instantiated as specified by its **RunOn** property. The instantiation might occur in the **main** method.
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>UpdatePermissions</td>
<td>Specify whether update permission should be available for section when</td>
</tr>
<tr>
<td></td>
<td>privileges are assigned to the menu item. The following options are available:</td>
</tr>
<tr>
<td></td>
<td>- <strong>Auto</strong> – The permission will be available for section as a privilege on</td>
</tr>
<tr>
<td></td>
<td>this menu item's <strong>Privileges</strong> node under the <strong>Entry Points</strong> node.</td>
</tr>
<tr>
<td></td>
<td>- <strong>No</strong> – The permission won't be available for section as a privilege on</td>
</tr>
<tr>
<td></td>
<td>the menu item.</td>
</tr>
<tr>
<td></td>
<td>The default value is <strong>Auto</strong>.</td>
</tr>
<tr>
<td>Web</td>
<td>Specify the URL that is opened when you run the menu item. The value of</td>
</tr>
<tr>
<td></td>
<td>this property is no longer used. Don't use this property.</td>
</tr>
<tr>
<td>WebConfigurationKey</td>
<td>Optional: Select a web-specific configuration key in addition to a</td>
</tr>
<tr>
<td></td>
<td>standard configuration key. This property applies to web menu items only.</td>
</tr>
<tr>
<td>WebMenuItemName</td>
<td>Specify the menu item to include on a web menu. The available values</td>
</tr>
<tr>
<td></td>
<td>depend on the setting of the <strong>WebMenuItemType</strong> property.</td>
</tr>
<tr>
<td>WebMenuItemType</td>
<td>Specify the type of the web menu item. There are two categories of web menu</td>
</tr>
<tr>
<td></td>
<td>items:</td>
</tr>
<tr>
<td></td>
<td>- <strong>URL</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Action</strong></td>
</tr>
<tr>
<td></td>
<td>The value that you select determines the web menu item names that are</td>
</tr>
<tr>
<td></td>
<td>available for the <strong>WebMenuItemName</strong> property.</td>
</tr>
<tr>
<td>WebPage</td>
<td>Specify the webpage that is linked to the menu item. The value of this</td>
</tr>
<tr>
<td></td>
<td>property is no longer used. Don't use this property.</td>
</tr>
<tr>
<td>WebSecureTransaction</td>
<td>Select whether the menu item requires secure transactions (SSL). This</td>
</tr>
<tr>
<td></td>
<td>property applies to web menu items only.</td>
</tr>
</tbody>
</table>

**NOTE**

When you use the **Parameters** or **EnumParameter** property, errors such as type mismatches can be found only at run time, not at compile time.

## Query properties

Within a query, you can set properties on the query itself, the data sources, the fields that are used for sorting, and the ranges that are used to delimit the query.

**Query properties**

Query properties determine the overall behavior of the query. For example, you can specify the form that is shown to users so that they can interact with the query.
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowCheck</td>
<td>This property is ignored for queries. It's effective on forms and reports.</td>
</tr>
<tr>
<td>AllowCrossCompany</td>
<td>Specify whether data is retrieved for all companies that the user has authority to read from. If the property is set to false, which is the default value, data is retrieved only for the current session company.</td>
</tr>
<tr>
<td>Description</td>
<td>Optional: Describe the query, what it returns, and so on. This property is useful in Microsoft Office Add-in scenarios.</td>
</tr>
<tr>
<td>Form</td>
<td>Specify that query form that MorphX should show when users interact with the query. The default value is SysQueryForm.</td>
</tr>
<tr>
<td>Interactive</td>
<td>Specify whether users can interact with the report by delimiting queries, setting printer options, and so on.</td>
</tr>
<tr>
<td>Literals</td>
<td>Specify how literals are represented in SQL statements. The forceLiterals option instructs the kernel to reveal the actual values that are used in where clauses to the Microsoft SQL Server database at the time of optimization. The forcePlaceholders option instructs the kernel not to reveal the actual values. Note: We don't recommend that you use the forceLiterals option, because it could expose code to an SQL injection security threat.</td>
</tr>
<tr>
<td>Model</td>
<td>Specify the model that the query is in. A model is a logical grouping of elements in a layer. Examples of elements include a table or class. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in a higher layer.</td>
</tr>
<tr>
<td>QueryType</td>
<td>Specify the type of the query. The following options are available:</td>
</tr>
<tr>
<td></td>
<td>• Join</td>
</tr>
<tr>
<td></td>
<td>• Union</td>
</tr>
<tr>
<td></td>
<td>The default value is Join.</td>
</tr>
<tr>
<td>Searchable</td>
<td>Specify whether the query can be part of a set of queries that is used to search the Microsoft SharePoint Business Catalog. This property is useful when you use the Enterprise Search feature. The default value is No.</td>
</tr>
<tr>
<td>Title</td>
<td>The heading for the query.</td>
</tr>
<tr>
<td>UserUpdate</td>
<td>Specify whether the query form should retain its state when it's reopened. If you set this property to Yes, the previous settings are restored. If you set it to No, the data can be viewed but not edited.</td>
</tr>
<tr>
<td>Version</td>
<td>The version is increased every time that the query is updated. This property is read-only.</td>
</tr>
</tbody>
</table>

**Data source properties**
The following properties control the characteristics of a data source. Additional properties are available on embedded data sources and relations between data sources. You can also set one property on fields in the data source.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>WHERE IT’S AVAILABLE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowAdd</td>
<td>Data source</td>
<td>Specify whether users can add fields to sorting and ranges at run time.</td>
</tr>
<tr>
<td>Company</td>
<td>Data source</td>
<td>Specify the company to retrieve data from.</td>
</tr>
<tr>
<td>Dynamic</td>
<td>Fields node in a data source</td>
<td>Specify whether all fields in the table in the data source are used. If you set this property to Yes, all the fields in the data source are used. If you set it to No, you can remove some of the fields. When the data source is a base table, a value of Yes means that all fields from the derived tables are used.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Data source</td>
<td>If you set this property to No, the data source (and all embedded data sources) are ignored.</td>
</tr>
<tr>
<td>FetchMode</td>
<td>Embedded data source</td>
<td>Specify whether the data sources should be related through a 1:1 relation or a 1:n relation. <strong>Note:</strong> For data sources that are used on reports, use a join relation that uses 1:1 fetch mode.</td>
</tr>
<tr>
<td>Field, RelatedField</td>
<td>Relations on an embedded data source</td>
<td>The name of the fields from the parent data source and related data source that are used in the relation.</td>
</tr>
<tr>
<td>FirstFast</td>
<td>Data source</td>
<td>If you set this property to Yes, the database receives a hint that the first record from the query should be retrieved before the other records. This setting enables some database systems to optimize record retrieval and therefore helps improve performance.</td>
</tr>
<tr>
<td>FirstOnly</td>
<td>Data source</td>
<td>If you set this property to Yes, the database receives a hint that only the first record from the query is required. This setting enables some database systems to optimize record retrieval and therefore helps improve performance.</td>
</tr>
<tr>
<td>JoinMode</td>
<td>Embedded data source</td>
<td>Specify the strategy that is used to join the output from a data source.</td>
</tr>
<tr>
<td>Name</td>
<td>Data source</td>
<td>Specify the name of the data source.</td>
</tr>
</tbody>
</table>
Relations
Embedded data source
Specify whether the query system should use the relations that are defined for tables and EDTs. If you set this property to Yes, the query is automatically updated if a relation is changed.

Table
Data source
Specify the table, map, or view that is used as a data source. This property can't be modified after a sorting order or a range has been defined.

Table, RelatedTable
Relations on an embedded data source
The name of the parent data source and the related data source.

Uniqueld
Data source
The unique number of the data source. This property is read-only.

Update
Data source
Specify whether the query can update records in the database.

Range properties
The following properties determine the characteristics of the range specification. For example, you can specify whether users are allowed to modify the range at run time.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>Use this property to disable a field in a range specification.</td>
</tr>
<tr>
<td>Field</td>
<td>Specify the field to define a range on.</td>
</tr>
<tr>
<td>Label</td>
<td>Enter a label for the range.</td>
</tr>
<tr>
<td>Status</td>
<td>Specify whether users are allowed to modify the range in the query dialog box at run time. The following options are available: • Open – Users can view and edit the range. • Lock – Users can only view the range. • Hide – Users can't view or edit the range.</td>
</tr>
<tr>
<td>Value</td>
<td>Specify the range for the records that are retrieved. If you use enumerations, don't use text strings. The enumeration ID must be used.</td>
</tr>
</tbody>
</table>

Report properties
Most of the properties for a report are set on the design, design section, and control nodes in Application Explorer. For information about system properties that are available on reports, see the “System and common properties” section. The following table describes the properties of a report.
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowCheck</td>
<td>Specify whether a message is shown when users try to run reports that they don't have permission to view. Select Yes to specify that a message is shown.</td>
</tr>
<tr>
<td>AutoJoin</td>
<td>Specify whether a record that is returned by the <code>element.args</code> method is used to set the range on the report query.</td>
</tr>
<tr>
<td>Interactive</td>
<td>Specify whether users can select which records to show by modifying the query that is associated with a report.</td>
</tr>
<tr>
<td>Model</td>
<td>Specify the model that the report is in. A model is a logical grouping of elements in a layer. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in another layer.</td>
</tr>
</tbody>
</table>

### Report control properties

The following table describes report control properties. For information about additional properties that are available for controls, see the "Form control properties" section.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment</td>
<td>Specify the alignment of a value that is shown in a control.</td>
</tr>
<tr>
<td>AllowNegative</td>
<td>Specify whether the control accepts negative values. This property is available only for integer and real controls.</td>
</tr>
<tr>
<td>ArrayIndex</td>
<td>Specify the array element that is shown in a control. The control is based on an extended data type that has array elements. This property isn't available for text and shape controls.</td>
</tr>
<tr>
<td>AutoDeclaration</td>
<td>Specify whether a variable is created that has the same name as the control. When you set this property to Yes, it's easier to access the report controls from X++ code, and you can find errors at compile time.</td>
</tr>
<tr>
<td>AutoInsSeparator</td>
<td>Specify whether a decimal separator is shown. This property is available only for real controls.</td>
</tr>
<tr>
<td>BackgroundColor</td>
<td>Specify the background color for a control. The setting of this property can be overridden by using the BackStyle property.</td>
</tr>
</tbody>
</table>
| BackStyle   | Specify whether the control background is opaque or transparent. When you set this property to Transparent, the behavior depends on the type of control:  
  - For bitmap controls, pixels that have the same color are transparent.  
  - For all other controls, the foreground color is used as the background color. |
<p>| Bold        | Specify bold text formatting. |</p>
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BottomMargin</td>
<td>Specify the margin for a control.</td>
</tr>
<tr>
<td>ChangeCase</td>
<td>Specify the case of text that a user enters. This property is available only for string, enum, text, and prompt controls.</td>
</tr>
</tbody>
</table>
| ChangeLabelCase       | Specify whether the label for the control should be modified when the report is printed. The following options are available:  
  • Auto  
  • None  
  • UPPER CASE  
  • lower case  
  • Title Case  
  The default value is Auto. |
<p>| ColorScheme           | Specify the color palette for a control.              |
| ConfigurationKey      | Specify a configuration key for the control.          |
| CSSClass              | Specify the Cascading Style Sheet (CSS) to use to render the value in HTML. |
| DataField             | Specify a table field for the control. This property isn't available for text, shape, box, and bitmap controls. |
| DataMethod            | Specify a display method that shows data in a control. This property isn't available for text, shape, and box controls. |
| DateDay               | Specify the format for the day. This property is available only for date controls. |
| DateFormat            | Specify the format for a date. This property is available only for date controls. |
| DateMonth             | Specify the format for the month. This property is available only for date controls. |
| DateSeparator         | Specify the separator that appears between the month, day, and year. This property is available only for date controls. |
| DateYear              | Specify the format for the year. This property is available only for date controls. |
| DecimalSeparator      | Specify the symbol that is used to separate decimal values. This property is available only for real controls. |
| DisplaceNegative      | Specify a new position for a value in a grid control when the value is a negative number. This property is available only for integer and real controls. |</p>
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>DynamicHeight</td>
<td>Specify whether the control is resized to show additional lines of text. When you set this property to Yes, page headers, page footers, and repeating column headings are automatically added as required. This property is available only for string controls.</td>
</tr>
<tr>
<td>ExtendedDataType</td>
<td>Specify the EDT that the field that is associated with the control should be based on.</td>
</tr>
<tr>
<td>ExtraSumWidth</td>
<td>Modify the default width that is allowed for sums. This property is available only for integer and real controls.</td>
</tr>
<tr>
<td>Font</td>
<td>Specify the font.</td>
</tr>
<tr>
<td>FontSize</td>
<td>Specify the font size.</td>
</tr>
<tr>
<td>ForegroundColor</td>
<td>Specify the foreground color for a control.</td>
</tr>
<tr>
<td>FormatMST</td>
<td>Specify whether values are formatted by using standard currency format. This property is available only for real controls.</td>
</tr>
<tr>
<td>Height</td>
<td>Specify the height of a control. When a control is associated with an EDT, the Height property of the control overrides the DisplayLength property of the EDT. If you set the Height property to Auto for a bitmap control, the size of the control is based on the size of the graphic.</td>
</tr>
<tr>
<td>ImageName</td>
<td>Specify the file name for an image. This property is available only for bitmap controls.</td>
</tr>
<tr>
<td>ImageResource</td>
<td>Specify the ID for a system resource to show. The resource macro provides a list of these IDs. Macros are located under the Macros node in Application Explorer. This property is available only for bitmap controls.</td>
</tr>
<tr>
<td>Italic</td>
<td>Specify italic text formatting.</td>
</tr>
<tr>
<td>Label</td>
<td>Specify a title for the control. If a label isn't specified here, it's inherited from the field.</td>
</tr>
<tr>
<td>LabelBold</td>
<td>Set or return a value that indicates the bold setting for the label in the control.</td>
</tr>
<tr>
<td>LabelCSSClass</td>
<td>Specify the CSS to use to render the label in HTML.</td>
</tr>
<tr>
<td>LabelFont</td>
<td>Set or return a string data type value that indicates the font for the label text in a form combo box control.</td>
</tr>
<tr>
<td>LabelFontSize</td>
<td>Set or return the font size, in points, for the label text in a form combo box control.</td>
</tr>
<tr>
<td>LabelItalic</td>
<td>Set or return a value that indicates whether the text in the control label should be italic.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LabelLineBelow</td>
<td>Specify the format of the underline for the control title.</td>
</tr>
<tr>
<td>LabelLineThickness</td>
<td>Specify the format of the line below column headings.</td>
</tr>
<tr>
<td>LabelPosition</td>
<td>Set or return the position of the label for the control. Valid values are <strong>Left</strong> and <strong>Above</strong>.</td>
</tr>
</tbody>
</table>
| LabelTabLeader   | Specify whether to append a series of dots to control labels. The following options are available:  
|                  |   • Auto  
|                  |   • Do not append  
|                  |   • Do append  
<p>|                  | The default value is <strong>Auto</strong>.                                             |
| LabelUnderline   | Set or return a value that indicates whether the text in the control label should be underlined. |
| LabelWidth       | Specify the width of the label for the control.                            |
| Left             | Specify the left alignment of a control.                                   |
| LeftMargin       | Specify the left margin for a control.                                     |
| Line             | Specify the appearance of the lines that form a shape. This property is available only for shape controls. |
| LineAbove        | Specify the type of line for the top border of a control. If your report has many lines or boxes, consider using a shape control inside the individual sections. |
| LineBelow        | Specify the type of line for the bottom border of a control. If your report has many lines or boxes, consider using a shape control inside the individual sections. |
| LineLeft         | Specify the type of line for the left border of a control. If your report has many lines or boxes, consider using a shape control inside the individual sections. |
| LineRight        | Specify the type of line for the right border of a control. If your report has many lines or boxes, consider using a shape control inside the individual sections. |
| MenuItemLabel    | Specify the label for a menu item.                                         |
| MenuItemName     | Specify the name of the menu item. The available menu items vary, depending on the setting of the <strong>MenuItemType</strong> property. |
| MenuItemType     | Specify whether the menu item is an action, display, or output menu item. A display menu item is for a form, and an output menu item is for a report. An output menu item is for a class, job, or query. |</p>
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>MinNoOfDecimals</td>
<td>Specify the minimum number of decimal places that are shown. Trailing zeros aren't shown.</td>
</tr>
<tr>
<td>ModelFieldName</td>
<td>Specify a field that is used to determine the left alignment and width of a control.</td>
</tr>
<tr>
<td>NoOfDecimals</td>
<td>Specify the number of decimal places that are shown. The default value is 20. This property is available only for real controls.</td>
</tr>
<tr>
<td>ResizeBitmap</td>
<td>Specify whether an image can be resized to fit the dimensions of a control. This property is available only for bitmap controls.</td>
</tr>
<tr>
<td>RightMargin</td>
<td>Specify the margin for a control.</td>
</tr>
<tr>
<td>RotateSign</td>
<td>Specify whether the sign for the control is inverted. This property is available only for integer and real controls.</td>
</tr>
<tr>
<td>ShowLabel</td>
<td>Set or return a value that indicates whether the label for the control is shown in the form. A value of True indicates that the label will be shown.</td>
</tr>
<tr>
<td>ShowPicAsText</td>
<td>Specify whether the file name for an image is shown instead of the image. This property is available only for bitmap controls.</td>
</tr>
<tr>
<td>ShowZero</td>
<td>Specify whether a 0 (zero) value is shown. This property is available only for integer and real controls.</td>
</tr>
<tr>
<td>SignDisplay</td>
<td>Specify how the sign of a number is shown. This property is available only for integer and real controls.</td>
</tr>
<tr>
<td>SumAll</td>
<td>Specify whether the sum of all values is calculated. This property is available only for integer and real controls.</td>
</tr>
<tr>
<td>SumNeg</td>
<td>Specify whether the sum of all negative values is calculated. This property is available only for integer and real controls.</td>
</tr>
<tr>
<td>SumPos</td>
<td>Specify whether the sum of all positive values is calculated. This property is available only for integer and real controls.</td>
</tr>
<tr>
<td>Table</td>
<td>Specify a data source for the control. This property isn't available for text, shape, box, and bitmap controls.</td>
</tr>
<tr>
<td>Text</td>
<td>Specify the text string that is shown in a control. This property is available only for text controls.</td>
</tr>
<tr>
<td>TimeFormat</td>
<td>Specify whether times are shown in 24-hour format or AM/PM format. This property is available only for time controls.</td>
</tr>
<tr>
<td>TimeHours</td>
<td>Specify whether hours are shown. This property is available only for time controls.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TimeMinutes</td>
<td>Specify whether minutes are shown. This property is available only for time controls.</td>
</tr>
<tr>
<td>TimeSeconds</td>
<td>Specify whether seconds are shown. This property is available only for time controls.</td>
</tr>
<tr>
<td>TimeSeparator</td>
<td>Specify the symbol that is used to separate hours, minutes, and seconds. This property is available only for time controls.</td>
</tr>
<tr>
<td>Thickness</td>
<td>Specify the thickness of a control border.</td>
</tr>
<tr>
<td>ThousandSeparator</td>
<td>Specify the symbol that is used to separate thousands. This property is available only for real controls.</td>
</tr>
<tr>
<td>Top</td>
<td>Specify the top alignment of a control.</td>
</tr>
<tr>
<td>TopMargin</td>
<td>Specify the margin for a control.</td>
</tr>
<tr>
<td>Type</td>
<td>Specify the type of shape that is shown. This property is available only for shape controls.</td>
</tr>
<tr>
<td>TypeHeaderPrompt</td>
<td>Specify whether a line of dots is added to fill the space between the control title and the control value. This property is available only for text and prompt controls.</td>
</tr>
<tr>
<td>Underline</td>
<td>Specify underline text formatting.</td>
</tr>
<tr>
<td>Visible</td>
<td>Set or return a value that indicates whether the control is visible. A value of True indicates that the control is visible.</td>
</tr>
<tr>
<td>WarnIfMissing</td>
<td>Specify whether a message is shown if an image is missing from the report. This property is available only for bitmap controls.</td>
</tr>
<tr>
<td>WebMenuItemName</td>
<td>Specify the menu item to include on a web menu. The available values depend on the setting of the WebMenuItemType property.</td>
</tr>
<tr>
<td>WebMenuItemType</td>
<td>Specify the type of the menu item. There are two categories of web menu items:</td>
</tr>
<tr>
<td></td>
<td>• URL</td>
</tr>
<tr>
<td></td>
<td>• Action</td>
</tr>
<tr>
<td></td>
<td>The value that you select determines the web menu item names that are available for the WebMenuItemName property.</td>
</tr>
<tr>
<td>WebTarget</td>
<td>Specify the location of the control on a web report.</td>
</tr>
<tr>
<td>Width</td>
<td>Specify the width of a control. When a control is associated with an EDT, the Width property of the control overrides the DisplayLength property of the EDT. If you set the Width property to Auto for a bitmap control, the size of the control is based on the size of the graphic.</td>
</tr>
</tbody>
</table>
# Report design properties

The following table describes the report design properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrangeMethod</td>
<td>Specify the layout for the controls in a report section.</td>
</tr>
<tr>
<td>ArrangeWhen</td>
<td>Specify when report controls are arranged.</td>
</tr>
<tr>
<td>BottomMargin</td>
<td>Specify the bottom margin. If you set this property to <strong>Auto</strong>, the default value that is stored in the system table is used.</td>
</tr>
<tr>
<td>Caption</td>
<td>Specify the name that is shown for the report in the user interface.</td>
</tr>
<tr>
<td>ColorScheme</td>
<td>Specify the color palette.</td>
</tr>
<tr>
<td>Columns</td>
<td>Specify the number of columns.</td>
</tr>
<tr>
<td>ColumnSpace</td>
<td>Specify the space between columns.</td>
</tr>
<tr>
<td>Font, FontSize, Italic, Underline and Bold</td>
<td>Specify the text formatting. The settings of the <code>Font</code> and <code>FontSize</code> properties override the values that you can set by clicking <strong>Options &gt; Fonts</strong> on the <strong>Tools</strong> menu.</td>
</tr>
<tr>
<td>ForegroundColor</td>
<td>Specify the foreground color.</td>
</tr>
<tr>
<td>Height</td>
<td>Specify the height.</td>
</tr>
<tr>
<td>LeftMargin</td>
<td>Specify the left margin. If you set this property to <strong>Auto</strong>, the default value that is stored in the system table is used.</td>
</tr>
<tr>
<td>LineAbove</td>
<td>Specify the type of line for the top border of a section.</td>
</tr>
<tr>
<td>LineBelow</td>
<td>Specify the type of line for the bottom border of a section.</td>
</tr>
<tr>
<td>LineLeft</td>
<td>Specify the type of line for the left border of a section.</td>
</tr>
<tr>
<td>LineRight</td>
<td>Specify the type of line for the right border of a section.</td>
</tr>
<tr>
<td>ResolutionX, ResolutionY</td>
<td>Specify the distance between gridlines.</td>
</tr>
<tr>
<td>RightMargin</td>
<td>Specify the right margin. If you set this property to <strong>Auto</strong>, the default value that is stored in the system table is used.</td>
</tr>
</tbody>
</table>
### Ruler
Specify the unit for the ruler that is shown when you edit a design. To edit a design, right-click **AutoDesignSpecs** or **Generated Design**, and then click **Edit**.

### Thickness
Specify the thickness of a section border.

### TopMargin
Specify the top margin. If you set this property to **Auto**, the default value that is stored in the system table is used.

## Report design section properties

The following table describes properties for report design sections. For information about additional properties that are available for report designs, see the “Report design properties” section.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrangeMethod</td>
<td>Specify the layout for the controls in a report section.</td>
</tr>
<tr>
<td>ArrangeWhen</td>
<td>Specify when the controls in the container should be arranged. The available options are <strong>Startup</strong>, <strong>On demand</strong>, and <strong>Never</strong>.</td>
</tr>
<tr>
<td>Bold</td>
<td>Get or set the weight of the font that was used to show text in the control.</td>
</tr>
<tr>
<td>Bottom</td>
<td>Change the position of the bottom of the report.</td>
</tr>
<tr>
<td>BottomMargin</td>
<td>Specify the bottom margin. If you set this property to <strong>Auto</strong>, the default value that is stored in the User Info system table is used.</td>
</tr>
<tr>
<td>ColorScheme</td>
<td>Specify the color palette.</td>
</tr>
<tr>
<td>ColumnHeadingsStrategy</td>
<td>Specify the layout of column headings. If you set this property to <strong>WordWrap</strong>, a heading wraps when it's longer than the longest field in the column. Headings can wrap to a maximum of eight lines. Headings that are longer than eight lines are truncated. <strong>Note</strong>: The heading length varies, depending on the language.</td>
</tr>
<tr>
<td>Columns</td>
<td>Specify the number of columns.</td>
</tr>
<tr>
<td>Columnspace</td>
<td>Specify the space between columns.</td>
</tr>
<tr>
<td>Font</td>
<td>Specify the text formatting. The settings of the <strong>Font</strong> and <strong>FontSize</strong> properties override the values that you can set by clicking <strong>Options &gt; Fonts</strong> on the <strong>Tools</strong> menu.</td>
</tr>
<tr>
<td>FontSize</td>
<td>Specify the text formatting. The settings of the <strong>Font</strong> and <strong>FontSize</strong> properties override the values that you can set by clicking <strong>Options &gt; Fonts</strong> on the <strong>Tools</strong> menu.</td>
</tr>
<tr>
<td>ForegroundColor</td>
<td>Specify the foreground color.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GrandHeader</td>
<td>Specify whether the value of the <strong>HeaderText</strong> property is shown. The <strong>GrandHeader</strong> property is available only when a report has multiple data sources that aren't nested.</td>
</tr>
<tr>
<td>GrandTotal</td>
<td>Specify whether the value of the <strong>FooterText</strong> property is shown. The <strong>GrandTotal</strong> property is available only when a report has multiple data sources that aren't nested.</td>
</tr>
<tr>
<td>HeaderText</td>
<td>Specify the text that is shown above the first record in a section when the <strong>GrandHeader</strong> property is set to <strong>Yes</strong>. This property is available only when a report has multiple data sources that aren't nested.</td>
</tr>
<tr>
<td>Height</td>
<td>Specify the height.</td>
</tr>
<tr>
<td>Italic</td>
<td>Specify the text formatting. The settings of the <strong>Font</strong> and <strong>FontSize</strong> properties override the values that you can set by clicking <strong>Options &gt; Fonts</strong> on the <strong>Tools</strong> menu.</td>
</tr>
<tr>
<td>LabelTopMargin, LabelBottomMargin</td>
<td>Specify the margins above and below column headings.</td>
</tr>
<tr>
<td>LeftMargin</td>
<td>Specify the left margin. If you set this property to <strong>Auto</strong>, the default value that is stored in the UserInfo system table is used.</td>
</tr>
<tr>
<td>LineAbove, LineBelow, LineLeft, LineRight</td>
<td>Specify the type of line for a section border. If a report has many lines and boxes, consider using the shape control inside a section.</td>
</tr>
<tr>
<td>Map</td>
<td>Specify the map to use to show data. You can associate a map field with a field in one or more tables. This property lets you use the same field name to access fields that have different names in different tables.</td>
</tr>
<tr>
<td>NoOfHeadingLines</td>
<td>Specify the number of lines that are used to show column headings. If you set the property to <strong>0</strong> (zero), column headings aren't displayed. For reports that include several fields, increase the number of lines to make sure that all fields are shown.</td>
</tr>
<tr>
<td>RightMargin</td>
<td>Specify the right margin. If you set this property to <strong>Auto</strong>, the default value that is stored in the UserInfo system table is used.</td>
</tr>
<tr>
<td>ResolutionX</td>
<td>Specify the distance between gridlines.</td>
</tr>
<tr>
<td>ResolutionY</td>
<td>Specify the distance between gridlines.</td>
</tr>
<tr>
<td>Ruler</td>
<td>Specify the unit for the ruler that is shown when you edit a design. To edit a design, right-click <strong>AutoDesignSpecs</strong> or <strong>Generated Design</strong>, and then click <strong>Edit</strong>.</td>
</tr>
<tr>
<td>Table</td>
<td>Specify the data source for a section.</td>
</tr>
<tr>
<td>Thickness</td>
<td>Specify the thickness of a section border.</td>
</tr>
</tbody>
</table>
### Top
Change the position of the top of the report.

### TopMargin
Specify the top margin. If you set this property to Auto, the default value that is stored in the UserInfo system table is used.

### Underline
Specify the text formatting. The settings of the Font and FontSize properties override the values that you can set by clicking Options > Fonts on the Tools menu.

## Report query properties

The following table describes report query properties. For information about additional report properties, see the "Report properties" and "System and common properties" sections.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowCheck</td>
<td>Get or set the Allow check flag.</td>
</tr>
<tr>
<td>AllowCrossCompany</td>
<td>Get or set the Allow cross-company flag. This flag indicates whether query execution will be across companies.</td>
</tr>
<tr>
<td>Description</td>
<td>A textual explanation of the query. This optional property is often used in Office Add-ins scenarios.</td>
</tr>
<tr>
<td>Form</td>
<td>Specify the form that is used for user interaction.</td>
</tr>
<tr>
<td>Interactive</td>
<td>Specify whether users can interact with the report by delimiting queries, setting printer options, and so on.</td>
</tr>
<tr>
<td>Literals</td>
<td>Specify how literals are represented in SQL statements.</td>
</tr>
<tr>
<td>Model</td>
<td>Specify the model that the report query is in. A model is a logical grouping of elements in a layer. Examples of elements include a table or class. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in another layer.</td>
</tr>
</tbody>
</table>
| QueryType    | Specify the type of the query. The following options are available:  
   - Join  
   - Union  
   The default value is Join. |
<p>| Searchable   | Specify whether the query can be part of a set of queries that can be used to search the SharePoint Business Catalog. This property is useful when you use the Enterprise Search feature. The default value is No. |
| Title        | Specify the title of the query. |</p>
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserUpdate</td>
<td>Specify whether users can update a query.</td>
</tr>
<tr>
<td>Version</td>
<td>This is a read-only internal property.</td>
</tr>
</tbody>
</table>

**Security code permission properties**

A code permission is a group of permissions that are associated with a menu item or a service operation. When a security role has access to a menu item, it also has access to other Application Explorer items that are mentioned within the code permission for that menu item. The degree of access is controlled by the specific permissions that are defined under the code permission node.

**Securable objects**

Code permissions are used to give access to securable objects. The following list shows the hierarchy of code permission nodes in Application Explorer:

- Security
  - Code Permissions
    - YourCodePermission
      - Tables
      - Server Methods
      - Associated Objects
        - Forms
        - Web Controls
        - Reports

Code permissions can also override the access levels to securable objects under the Associated Objects node.

**Code permission properties**

This following table describes the properties for the node at Security > Code Permissions > YourCodePermission in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Yes</td>
<td>The name of the code permission. The code permission lets users run the class method that is specified in the Method property.</td>
</tr>
<tr>
<td>Class</td>
<td>Optional</td>
<td>The class that is associated with this code permission.</td>
</tr>
<tr>
<td>Method</td>
<td>Optional</td>
<td>The method that is associated with this code permission.</td>
</tr>
</tbody>
</table>

**Table properties**

The following table describes the properties for the node at Security > Code Permissions > YourCodePermission > Tables > YourTable in Application Explorer.
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Yes</td>
<td>The name of the table.</td>
</tr>
</tbody>
</table>
| EffectiveAccess | Yes    | The permission value. The following options are available:  
  - Read  
  - Update  
  - Create  
  - Correct  
  - Delete  
  - NoAccess  
  
The permission values for the `EffectiveAccess` property represent a hierarchy. Read is the weakest permission, and Delete is the strongest. Delete permission includes every other permission. Create permission includes Update and Read. You can set the permission value to `NoAccess` to prevent all access to the table. |
| ManagedBy     | Optional | This property is used by automation tools. |

**Server method properties**

The following tables describes the properties for the node at `Security > Code Permissions > YourCodePermission > Server Methods > YourServerMethod` in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Yes</td>
<td>The name of the server class.</td>
</tr>
<tr>
<td>Method</td>
<td>Yes</td>
<td>The secure server method that is tagged with the <code>SysEntryPointAttribute</code> attribute.</td>
</tr>
</tbody>
</table>
| EffectiveAccess | Yes    | The permission value. The following options are available:  
  - Invoke – The server method can be called.  
  - NoAccess – The server method can't be called. |
| ManagedBy     | Optional | This property is used by automation tools. |

**Form properties**

The following table describes the properties for the node at `Security > Code Permissions > YourCodePermission > Associated Objects > Forms > YourForm` in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form</td>
<td>Yes</td>
<td>The name of the form.</td>
</tr>
</tbody>
</table>
### Web control properties

The following table describes the properties for the node at `Security > Code Permissions > YourCodePermission > Associated Objects > Web Controls > YourWebControl` in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebControl</td>
<td>Yes</td>
<td>The name of the web control.</td>
</tr>
</tbody>
</table>
| AccessLevel| Yes      | The permission value. The following options are available:  
- Read  
- Update  
- Create  
- Correct  
- Delete  
- NoAccess  

The permission values for the `EffectiveAccess` property represent a hierarchy. Read is the weakest permission, and Delete is the strongest. Delete permission includes every other permission. Create permission includes Update and Read. You can set the permission value to `NoAccess` to prevent all access to the web control. |
| ManagedBy  | Optional | This property is used by automation tools. |

### Report properties

The following table describes the properties for the node at `Security > Code Permissions > YourCodePermission > Associated Objects > Reports > YourReport` in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ManagedBy</td>
<td>Optional</td>
<td>This property is used by automation tools.</td>
</tr>
</tbody>
</table>
Security duty properties

Security permissions are combined into privileges, and privileges are combined into duties. Duties are defined as groups of related privileges that provide a user with access to a specific business function. In Application Explorer, these privileges are organized into the nodes of a duty.

Best practices

This section describes the best practice rules for duties.

- All duties should be assigned to a role.
- All duties should be part of a process cycle.
- Because a duty represents a specific business function, the name of the duty should rarely or never change. For example, your company pays bills. Although the details of how you pay bills might change, the essential function of paying bills won't change. Instead of creating a new duty, you should change the privilege subnodes of the duty.
- The name of a process cycle should rarely or never change.

Duty hierarchy in Application Explorer

The following list shows the hierarchy of duty nodes in Application Explorer:

- Security
  - Duties
    - YourDuty
    - Privileges

Duty properties

The following table describes the properties for the node at Security > Duties > YourDuty in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Yes</td>
<td>The name of the duty.</td>
</tr>
<tr>
<td>Label</td>
<td>Yes</td>
<td>Text that is shown for the duty in the user interface.</td>
</tr>
<tr>
<td>Description</td>
<td>Yes</td>
<td>A description of the duty.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Yes</td>
<td>A value that indicates whether the duty is enabled. The following options</td>
</tr>
<tr>
<td></td>
<td></td>
<td>are available:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Yes – Enable the duty.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No  – Disable the duty.</td>
</tr>
</tbody>
</table>
Privilege properties
The following table describes the properties for the node at Security > Duties > YourDuty > Privileges > YourPrivilege in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Yes</td>
<td>The name of the privilege.</td>
</tr>
</tbody>
</table>
| Enabled    | Yes      | A value that indicates whether the duty is enabled. The following options are available:
|            |          | • Yes – Enable the privilege.                    |
|            |          | • No – Disable the privilege.                    |

Security privilege properties
A privilege is a group of permissions. The nodes that are below each privilege node identify the securable objects that a user can access and set the level of access for each object.

Best practices
This section describes the best practice rules for privileges.

- You can use privileges to specify the access that is required in order to perform a job.
- You can use privileges to group the permissions for related securable objects. For example, menu items and their controls are closely related.
- You can assign privileges directly to security roles. However, security settings are easier to maintain if you assign duties or process cycles instead of privileges.

Securable objects
Privileges are used to give access to securable objects. The following list shows the hierarchy under the Security > Privileges node in Application Explorer:

- Security
  - Privileges
    - YourPrivilege
      - Entry Points
      - Permissions
        - Tables
        - Server Methods
        - Forms

Privileges can also override the access levels to securable objects as they are defined elsewhere in Application Explorer. For example, a privilege can override a permission that is defined by the EffectiveAccess property under Forms > YourForm > Permissions > Update > Tables > YourTable in Application Explorer.

Privilege properties
The following table describes the properties for the node at Security > Privileges > YourPrivilege in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Yes</td>
<td>The name of the privilege.</td>
</tr>
</tbody>
</table>
### Privilege properties

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Yes</td>
<td>Text that is shown for the privilege in the user interface.</td>
</tr>
<tr>
<td>Description</td>
<td>Yes</td>
<td>A description of the privilege.</td>
</tr>
</tbody>
</table>
| Enabled            | Yes      | A value that indicates whether the duty is enabled. The following options are available:  
  * Yes – Enable the privilege.  
  * No – Disable the privilege. |

### Entry point properties

The following table describes the properties for the node at `Security > Privileges > YourPrivilege > Entry Points > YourEntryPoint` in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Yes</td>
<td>The name of the entry point.</td>
</tr>
</tbody>
</table>
| ObjectType         | Yes      | The object type of the entry point. The following options are available:  
  * MenuItemAtDisplay  
  * MenuItemAtOutput  
  * MenuItemAtAction  
  * ServiceOperation  
  * WebActionItem  
  * WebURLItem  
  * WebManagedContent |
| ObjectName         | Yes      | The object name of the entry point. |
| ObjectChildName    | Optional | A value that represents the service method name. **Note:** Specify a value for this property only if the `ObjectType` property is set to `ServiceOperation`. |
The permission value. For all object types except ServiceOperation, the following options are available:

- Read
- Update
- Create
- Correct
- Delete
- NoAccess

The permission values for the AccessLevel property represent a hierarchy. Read is the weakest permission, and Delete is the strongest. Delete permission includes every other permission. Create permission includes Update and Read. You can set the permission value to NoAccess to prevent all access to the entry point. The Correct permission applies only when a time state table is involved. This permission authorizes you to issue update records in a time state table. For the ServiceOperation object type, the following options are available:

- Invoke – The server method can be called.
- NoAccess – The server method can't be called.

Table properties

The following table describes the properties for the node at Security > Privileges > YourPrivilege > Permissions > Tables > YourTable in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Yes</td>
<td>The name of the table.</td>
</tr>
</tbody>
</table>
EffectiveAccess

The permission value. The following options are available:
- Read
- Update
- Create
- Correct
- Delete
- NoAccess

The permission values for the EffectiveAccess property represent a hierarchy. Read is the weakest permission, and Delete is the strongest. Delete permission includes every other permission. Create permission includes Update and Read. The Correct permission applies only when a time state table is involved. This permission authorizes you to update records in a time state table. You can set the permission value to NoAccess to prevent all access to the table.

ManagedBy

This property is used by automation tools.

Server method properties

The following table describes the properties for the node at Security > Privileges > YourPrivilege > Permissions > Server Methods > YourServerMethod in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Yes</td>
<td>The name of the server class.</td>
</tr>
<tr>
<td>Method</td>
<td>Yes</td>
<td>The name of the secure server method that is tagged with the SysEntryPointAttribute attribute.</td>
</tr>
</tbody>
</table>
| EffectiveAccess | Yes     | The permission value. The following options are available:
|               |          | • Invoke – The server method can be called.
|               |          | • NoAccess – The server method can't be called. |
| ManagedBy    | Optional | This property is used by automation tools. |

Form properties

The following table describes the properties for the node at Security > Privileges > YourPrivilege > Permissions > Forms > YourForm in Application Explorer.
Security process cycle properties

A process cycle is a group of duties. A process cycle represents a high-level job function. Although the details of how a given job function is run might change over time, the concept and name of that job function probably don't change.

Best practices

This section describes the best practice rules for process cycles.

- Each duty should be part of a process cycle.
- Use a process cycle to organize a group of duties for a job function.

Process cycle hierarchy in Application Explorer

The following list shows the hierarchy of process cycles nodes in Application Explorer:

- Security
  - Process Cycles
    - YourProcessCycle
      - Duties

Process cycle properties

The following table describes the properties for the node at Security > Process Cycles > YourProcessCycle in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Yes</td>
<td>The name of the process cycle.</td>
</tr>
<tr>
<td>Label</td>
<td>Yes</td>
<td>Text that is shown for the process cycle in the user interface.</td>
</tr>
<tr>
<td>Description</td>
<td>Yes</td>
<td>A description of the process cycle.</td>
</tr>
</tbody>
</table>
| Enabled    | Yes      | A value that indicates whether the duty is enabled. The following options are available:  
  - Yes – Enable the process cycle.  
  - No – Disable the process cycle. |

Duty properties

The following table describes the properties for the node at Security > Process Cycles > YourProcessCycle > Duties > YourDuty in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Yes</td>
<td>The name of the duty.</td>
</tr>
</tbody>
</table>
A value that indicates whether the duty is enabled. The following options are available:
- Yes – Enable the duty.
- No – Disable the duty.

Security policy properties
Developers and system administrators can create security policies to deny access to a subset of data records in tables.

Constrained tables of a policy
You can add tables and views under the Constrained Tables node of a security policy in Application Explorer. These tables and views are related to the data source table of the query that is named in the Query property of the policy. The following list shows the hierarchy of security policy nodes in Application Explorer:

- Security
  - Policies
    - YourPolicy
      - Constrained Tables
        - YourConstrainedTable
          - YourConstrainedSubTable
          - YourConstrainedView

Each Constrained Tables node can contain any number of constrained tables and views. Additionally, each constrained table can contain any number of constrained sub-tables.

Security policy properties
The following table describes the properties for the node at Security > Policies > YourPolicy in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Yes</td>
<td>The name of the security policy.</td>
</tr>
<tr>
<td>Label</td>
<td>Yes</td>
<td>The text that is shown for the security policy in the user interface.</td>
</tr>
<tr>
<td>PrimaryTable</td>
<td>Yes</td>
<td>The table that is specified in the data source for the security policy query.</td>
</tr>
<tr>
<td>Query</td>
<td>Yes</td>
<td>The query that the policy uses to filter data from the constrained tables that are specified in the policy.</td>
</tr>
<tr>
<td>UseNotExistJoin</td>
<td>Yes</td>
<td>A value that indicates whether the security query must be applied as a not exists join or an exists join.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>REQUIRED</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PolicyGroup</td>
<td>No</td>
<td>Administrators and developers can use this property to quickly identify groups of related security policies. The available options are the names of the security policy groups that the system administrator or developer has created. The system doesn't use this property at run time.</td>
</tr>
<tr>
<td>ConstrainedTable</td>
<td>Yes</td>
<td>A value that controls whether the security policy restricts the data values in records that are returned from the primary table. The following options are available:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Yes – The security policy is enforced on the primary table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• No – The security policy isn’t enforced on the primary table.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Yes</td>
<td>A value that controls whether the system enforces the policy at run time. The following options are available:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Yes – Enable the security policy.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• No – Disable the security policy.</td>
</tr>
<tr>
<td>Operation</td>
<td>Yes</td>
<td>A value that controls which data operations the policy is enforced for. The following options are available:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Select</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Insert</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Update</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Delete</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Insert, Update and Delete</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• All operations</td>
</tr>
<tr>
<td>ContextType</td>
<td>Yes</td>
<td>A value that controls the context type of the security policy. The following options are available:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ContextString – You must specify a value for the ContextString property. The security policy uses a specific application context for the policy.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• RoleName – The security policy is applied only to the application user who is assigned to the value of RoleName.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• RoleProperty – This value is used in combination with the ContextString property to specify multiple roles context.</td>
</tr>
</tbody>
</table>
Security role properties

Roles represent a collection of permissions that can be granted to users. The nodes that are nested below each role node identify various securable objects that a user can access and specify the level of access.

Role node in Application Explorer

Roles are used to give access to securable objects. The following list shows the hierarchy of role nodes in Application Explorer:

- Security
  - Roles
    - YourRole
      - Duties
      - Privileges
      - Permissions
        - Tables
        - Forms
        - Server Methods
      - Sub Roles

Roles are typically associated with security duties, and sometimes with security privileges. Access levels to securable objects within a role are derived from the duties, privileges, or both. Roles can also override the access levels to securable objects under the Permissions node.

Role properties

The following table describes the properties for the node at Security > Roles > YourRole in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Yes</td>
<td>The name of the role.</td>
</tr>
<tr>
<td>Label</td>
<td>Yes</td>
<td>Text that is shown for the role in the user interface.</td>
</tr>
<tr>
<td>Description</td>
<td>Yes</td>
<td>A description of the role.</td>
</tr>
</tbody>
</table>
| Enabled      | Yes      | A value that indicates whether the duty is enabled. The following options are available:  
- **Yes** – Enable the role.  
- **No** – Disable the role. |
PastDataAccess

The past data access for the tables that have date-effective fields. The following options are available:
- Read
- Update
- Create
- Correct
- Delete
- NoAccess

The permission values for the PastDataAccess property represent a hierarchy. Read is the weakest permission, and Delete is the strongest. Delete permission includes every other permission. Create permission includes Update and Read. You can set the permission value to NoAccess to prevent all access to the table.

CurrentDataAccess

The current data access for the tables that have date-effective fields.

FutureDataAccess

The future data access for the tables that have date-effective fields.

ContextString

A user-defined string that can be used by security policies.

**Duty properties**

The following table describes the properties for the node at Security > Roles > Duties > YourDuty in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Yes</td>
<td>The name of the duty.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Yes</td>
<td>A value that indicates whether the duty is enabled. The following options are available:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Yes – Enable the duty.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- No – Disable the duty.</td>
</tr>
</tbody>
</table>

**Privilege properties**

The following table describes the properties for the node at Security > Roles > Privileges > YourPrivilege in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Yes</td>
<td>The name of the privilege.</td>
</tr>
</tbody>
</table>
### Table properties

The following table describes the properties for the node at Security > Roles > Permissions > Tables > YourTable in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| Enabled          | Yes      | A value that indicates whether the duty is enabled. The following options are available:  
                    - Yes – Enable the privilege.  
                    - No – Disable the privilege. |

#### Table properties

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Yes</td>
<td>The name of the table.</td>
</tr>
</tbody>
</table>
| EffectiveAccess | Yes    | The permission value. The following options are available:  
                        - Read  
                        - Update  
                        - Create  
                        - Correct  
                        - Delete  
                        - NoAccess  
                        The permission values for the EffectiveAccess property represent a hierarchy. Read is the weakest permission, and Delete is the strongest. Delete permission includes every other permission. Create permission includes Update and Read. You can set the permission value to NoAccess to prevent all access to the table. |
| ManagedBy     | Optional | This property is used by automation tools. |

#### Form properties

The following table describes the properties for the node at Security > Roles > Permissions > Form > YourForm in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form</td>
<td>Yes</td>
<td>The name of the form.</td>
</tr>
</tbody>
</table>

#### Server method properties

The following table describes the properties for the node at Security > Roles > Permissions > Server Methods > YourServerMethod in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Yes</td>
<td>The name of the server class.</td>
</tr>
</tbody>
</table>
Method

Yes

The name of the secure server method that is tagged with the SysEntryPointAttribute attribute.

EffectiveAccess

Yes

The permission value. The following options are available:
- **Invoke** – The server method can be called.
- **NoAccess** – The server method can't be called.

ManagedBy

Optional

This property is used by automation tools.

Subrole properties

The following table describes the properties for the node at Security > Roles > Sub Roles > YourSubRole in Application Explorer.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>REQUIRED</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Yes</td>
<td>The name of the subrole.</td>
</tr>
</tbody>
</table>
| Enabled          | Yes      | A value that indicates whether the duty is enabled. The following options are available:  
- **Yes** – Enable the subrole  
- **No** – Disable the subrole. |

Web menu properties

The following table describes the properties that are specific to web menus and submenus.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConfigurationKey</td>
<td>Specify the configuration key that is used to control display of this menu. If a user doesn't have access to the configuration key, the menu won't be visible.</td>
</tr>
<tr>
<td>HighlightSelected</td>
<td>This property isn't supported.</td>
</tr>
<tr>
<td>Label</td>
<td>Specify the text that is shown for the top-level node of the web menu or submenu. The value can't exceed 250 characters.</td>
</tr>
<tr>
<td>MenuItemName</td>
<td>Specify the menu item to access when the top-level node for the menu or submenu is clicked. The options that are available depend on the setting of the MenuItemType property.</td>
</tr>
<tr>
<td>MenuItemType</td>
<td>Specify the type of menu item that is accessed by the top-level node of the menu or submenu. The available options are Action and URL.</td>
</tr>
</tbody>
</table>
Model
Specify the model. A model is a logical grouping of elements in a layer. Examples of elements include a table or class. An element can exist in exactly one model in a layer. The same element can exist in a customized version in a model that is in a higher layer.

SetCompany
This property causes the system to change the company when the form receives focus. If the *SaveDataPerCompany* property on a table is set to *Yes*, the *SetCompany* property on a form design that uses the table as a data source must also be set to *Yes*.

ShowParentModule
Specify whether to update the QuickLaunch, based on the parent module of the menu item. The following options are available:
- **Yes** – Always update the QuickLaunch, based on the parent module of the menu item.
- **No** – Leave the QuickLaunch unchanged, even if the parent module of the menu item differs from the current module.

The default value is **Yes**.

### Web menu item properties

The following table describes the properties that are specific to web menu items.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Big</strong></td>
<td>Specify the size of the button when it's used on an Action Pane. The following options are available:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Yes</strong> – The button is shown at full size and is located at the start of the group.</td>
</tr>
<tr>
<td></td>
<td>• <strong>No</strong> – The button is shown at the smaller size and is located on the right side of the group.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CloseDialogBehavior</th>
<th>Specify the action that is performed on the parent window when the dialog box closes. The following options are available:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Auto</strong> – Depending on how the dialog box was used, the appropriate update actions are performed when the dialog box is closed.</td>
</tr>
<tr>
<td></td>
<td>• <strong>RefreshDataSource</strong> – The read-only data source on the parent form is updated. This option preserves the current selection and performs a <em>Research()</em> operation on the data source.</td>
</tr>
<tr>
<td></td>
<td>• <strong>RefreshPage</strong> – Refresh the page.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Submit</strong> – Refresh the parent page.</td>
</tr>
<tr>
<td></td>
<td>• <strong>None</strong> – No action is performed.</td>
</tr>
<tr>
<td></td>
<td>The default value is <strong>Auto</strong>.</td>
</tr>
</tbody>
</table>

<p>| HideActionPane            | Specify whether the Action Pane is visible on the page that is being opened.                                               |</p>
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>HomePage</td>
<td>Specify whether the page is a Role Center page and is deployed to the main Enterprise Portal site.</td>
</tr>
<tr>
<td>NeedsRecord</td>
<td>When you set this property to Yes, the menu item is shown when there are no records in the data set.</td>
</tr>
<tr>
<td>PageDefinition</td>
<td>The page that the web menu item points to.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Specify the arguments that are passed to the page that is being opened. Each parameter must have the following form: \textit{name=value} if multiple parameters must be passed, they must be separated by an ampersand (&amp;), as shown in the following example: mode=2&amp;category=1</td>
</tr>
<tr>
<td>URL</td>
<td>Specify the URL to navigate to.</td>
</tr>
<tr>
<td>WebConfigurationKey</td>
<td>Select the configuration key that is required in order to enable the web menu item. Use the key for the module that the object belongs to.</td>
</tr>
<tr>
<td>WindowMode</td>
<td>Specify the type of window to use for the page that is being opened. The following options are available:</td>
</tr>
<tr>
<td></td>
<td>• \textit{Inline} – The page that is being opened will replace the existing content in the browser. If the web menu item is being accessed from a dialog box, the page that is being opened will open in a new browser window.</td>
</tr>
<tr>
<td></td>
<td>• \textit{Modal} – If no dialog box is open, a new dialog box will be created. If the web menu item is being accessed from a dialog box, the page that is being opened will replace the content of the current dialog box.</td>
</tr>
<tr>
<td></td>
<td>• \textit{NewModal} – The page that is being opened will always open in a new dialog box.</td>
</tr>
<tr>
<td></td>
<td>• \textit{NewWindow} – The page that is being opened will open in a new browser window.</td>
</tr>
<tr>
<td>WindowParameters</td>
<td>Specify additional parameters to control the appearance of the SharePoint dialog box. The parameters must be enclosed in braces ({}) and separated by commas. The following example shows how to set the \textit{WindowParameters} property so that the dialog box has a size of 400 × 300 pixels, and so that it has no \textit{Close} or \textit{Maximize} button: {width:400, height:300, showClose:false, allowMaximize:false}</td>
</tr>
<tr>
<td>WindowSize</td>
<td>Specify the size of the window to use for the page that is being opened. The following options are available:</td>
</tr>
<tr>
<td></td>
<td>• \textit{Smallest} – 330 × 200 pixels</td>
</tr>
<tr>
<td></td>
<td>• \textit{Small} – 550 × 450 pixels</td>
</tr>
<tr>
<td></td>
<td>• \textit{Medium} – 800 × 630 pixels</td>
</tr>
<tr>
<td></td>
<td>• \textit{Large} – 930 × 630 pixels</td>
</tr>
<tr>
<td></td>
<td>• \textit{Maximum} – The largest size that will fit in the boundaries of the main browser window</td>
</tr>
</tbody>
</table>
Usage

The `SysSetupConfigAttribute` attribute must be added for all X++ classes that implement the `SysSetup` interface. It accepts two parameters:

- **ContinueOnError** – This parameter is of the `bool` type. If execution of the X++ class fails during synchronization, database synchronization will either fail or continue with the next steps, depending on the value of this parameter (`true` or `false`).
  - `true` – Database synchronization will continue with the next steps.
  - `false` – The overall database synchronization operation will fail and can't be resumed until the underlying issue is fixed.

- **Timeout** – This parameter is of the `int` type, and the range of values is from 1 to 600 seconds. It defines the time range that the database synchronization operation will run the `SysSetup` class during.

In the following code example, the `ContinueOnError` parameter is set to `true`, and the `Timeout` parameter is set to `300`.

```plaintext
[SysSetupConfigAttribute(true, 300)]
class DemoClass implements SysSetup
{
    // Class code here.
}
```

**NOTE**

If the X++ class does not have the `SysSetupConfigAttribute` attribute, then default values are applied. `ContinueOnError` is `true` and `Timeout` is `120` seconds.
X++ is an object-oriented, application-aware, and data-aware programming language used in enterprise resource planning (ERP) programming and in database applications. It provides system classes for a broad range of system programming areas, highlighted in the following table.

<table>
<thead>
<tr>
<th>X++ Language Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes</td>
<td>In addition to system classes, there are also application classes for managing many types of business processes. Reflection on classes is supported.</td>
</tr>
<tr>
<td>Tables</td>
<td>X++ programmers can access the relational tables. X++ includes keywords that match most of the keywords in standard SQL. Reflection on tables is supported.</td>
</tr>
<tr>
<td>User interface</td>
<td>Manipulation of user interface items, such as forms and reports.</td>
</tr>
<tr>
<td>Best practice checks</td>
<td>X++ code is checked for syntax errors during compile time. The compile process also performs best practice checks. Violations of best practices can generate compiler messages.</td>
</tr>
<tr>
<td>Garbage collection</td>
<td>The X++ runtime execution engines have automatic mechanisms to discard objects that are no longer referenced, so that memory space can be reused.</td>
</tr>
<tr>
<td>Interoperability</td>
<td>Interoperability between classes written in X++ and in C# (or other .NET Framework languages) is supported.</td>
</tr>
<tr>
<td>File manipulation</td>
<td>File input and output are supported, including XML building and parsing.</td>
</tr>
<tr>
<td>Collections</td>
<td>Dynamic arrays are supported and the X++ includes several collection objects.</td>
</tr>
</tbody>
</table>

X++ compiles to Microsoft .NET CIL (Common Intermediate Language)

X++ source code is compiled to Microsoft .NET CIL (Common Intermediate Language). CIL is what the .NET compilers for C# and Visual Basic generate. The advantages of compiling to CIL include:

- Your code runs much faster than in previous versions (AX2012 and earlier).
- It’s easier to write application logic in other managed languages and integrated that logic into your X++ app.
- Your X++ apps can efficiently reference classes that are available in other .NET assembly DLL files.
- The CIL can be operated on by the many .NET tools.

The standard compilation unit is the same as for other .NET language. If any method in a model element (for example, a class, form, or query) fails to compile, the whole compilation fails.

If you are upgrading code from previous versions (AX2012 and earlier), note that the CIL helper methods such
as Global::runClassMethodIL have been removed, because they’re no longer relevant.

For more information, see What is "managed code"?

The Ignore list

Assemblies are generated from successful compilations and the runtime system can't load incomplete assemblies. There are scenarios when porting legacy applications where it's beneficial to get things running in a staged fashion and where parts of the application need to be tested before everything is ported. While this is useful for this very limited scenario, it shouldn't be used once the application is ready for production, since you would be hiding problems that will occur at runtime, after the system has been deployed. To ignore parts of your X++ code, you can specify a method in an XML by selecting "Edit Best Practice Suppressions," from the context menu on the project. This will open an XML document where the exclusions are maintained.

Concepts

The X++ language programming reference is divided into these sections:

- Variables and data types
- Statements, loops, and exception handling
- Operators
- Classes and methods
- Data selection and manipulation
- Macros
- Attribute classes

Additional resources

- X++ Syntax
- X++ and C# Comparison
This topic describes variables in X++.

- A **variable** is an identifier that points to a memory location where information of a specific data type is stored. The size, precision, default value, implicit and explicit conversion functions, and range depend on the variable's data type.
- The **scope** of a variable defines the area in the code where an item can be accessed.
- **Instance variables** are declared in class declarations, and can be accessed from any methods in the class or from methods that extend the class.
- **Local variables** can be accessed only in the block where they were defined.
- When a variable is declared, memory is allocated, and the variable is initialized to the default value.
- You can assign values to both static fields and instance fields as part of the declaration statement.
- Variables can be declared anywhere in a code block in a method. They don't have to be declared at the beginning of a method.
- **Constants** are variables where the value can't be changed when the variable is declared. They use the `const` or `readonly` keyword.
- Constants differ from read-only fields in only one way. Read-only fields can be assigned a value only one time, and that value never changes. The field can be assigned its value either inline, at the place where the field is declared, or in the constructor.

When you declare variables of managed types that aren't authored in X++, you have two options. You can fully qualify the type names in the declaration by including the full namespace, or you can add a `using` statement to your file and then omit the namespace from the type name.

### Variable examples

```x++
// An example of two valid variable names.
str variableName;
CustInfo custNumber;

// An example of simultaneously declaring and initializing a variable.
real pi = 3.14159265359; // Assigns value of pi to 12 significant digits.

// An example of initializing an object by using the new method on the class.
Access accessObject = new Access(); // Simple call to the new method in the Access class.

// An example of multiple declarations using integers.
int i,j; // Declares 2 integers, i and j.

// An example of multiple declarations using an array.
int a[100,5], b=1; // Declares array with 100 integers with 5 in memory and b as an integer with value 1.

// An example of how variable scopes work.
class ScopeExample
{

    // The variable a is declared within the class.
    int a;

    // Because the method below is declared within the class,
    // it can access all the variables defined within the class.
    void aNewMethod()
    {

    // The variable a is declared within the method.
    
```
The variable b is declared within the method. It can only be accessed by this method.

```csharp
int b;
}
}
```

An example of an assignment of field members inline.

```csharp
public class FieldAssignmentExample
{
    int field1 = 1;
    str field2 = "Banana";
    void new()
    {
        // ...
    }
}
```

class ConstantExample
{
    // An example of a constant being declared at the class level.
    public const str MyContent = 'SomeValue';
    public int ResultSoFar()
    {
        return 1;
    }
}

// The constants can then be referenced by using the double-colon syntax.
str value = ConstantExample::MyContent;
// If you're in the scope of the class where the const is defined,
// you can omit the type name prefix (ConstantExample in this example).

// An example of the using clause where the alias can denote
// namespaces and classes.
using System;
using IONS=System.IO; // Namespace alias.
using Alist=System.Collections.ArrayList; // Class alias.
public class NamespaceExample
{
    public static void Main(Args a)
    {
        Int32 I; // Alternative to System.Int32.
        Alist al; // Using a class alias.
        al = new Alist();
        str s;
        al.Add(1);
        IONS.Path::ChangeExtension(@"c:\tmp\test.xml", ".txt");
    }
}

var

You can declare a variable without explicitly providing the type of the variable, if the compiler can determine the type from the initialization expression. The variable is still strongly-typed into one, unambiguous type.

You can use var only on declarations where initialization expressions are provided. (The compiler will infer the type from the initialization expression.) In some cases, this approach can make code easier to read.

You should use var to declare local variables in these situations:

- When the type of the variable is obvious from the right side of the assignment
- When the exact type isn't important
- For the declarations of for loop counters
- For disposable objects inside using statements
Don't use `var` when the type isn't clear from the initialization expression.

**var examples**

```csharp
// When the type of a variable is clear from
// the context, use var in the declaration.
var var1 = "This is clearly a string.";
var var2 = 27; // This is an integer (not a real).
var i = System.Convert::ToInt32(3.4);

// Don't use var when the type of the variable is not clear
// from the context. Use an explicit type instead.
int var4 = myObject.ResultSoFar();
```

**Declare anywhere**

Declarations can now be provided wherever statements can be provided. A declaration is syntactically a statement, a *declaration statement*.

You can provide declarations immediately before the variable is used, and you don’t have to declare all the variables in one place. Therefore, you have precise control over the scope of your variables.

You can give variables smaller scopes, outside which the variables can’t be referenced. The lifetime of the variable is the scope that it’s declared in. Scopes can be started at the block level (inside compound statements), in `for` statements, and in `using` statements.

There are several advantages to making scopes small:

- Readability is enhanced.
- You reduce the risk that a variable will be reused in an inappropriate manner during long-term maintenance of the code.
- It’s easier to refactor code. You can copy in code without having to worry that variables might be reused in contexts where they shouldn’t be reused.

In the following example, we declare the loop counter inside the `for` statement that it's used in.

```csharp
void MyMethod()
{
    for (int i = 0; i < 10; i++)
    {
        info(strfmt("i is %1", i));
    }
}
```

The scope of the variable is the `for` statement itself, and includes the condition expression and the loop update parts. The value can’t be used outside this scope.

In the following example, when the compiler reaches the `info` statement, it will issue the following error message: "'i' isn't declared."
```csharp
void MyMethod()
{
    for (int i = 0; i < 10; i++)
    {
        if (i == 7)
        {
            break;
        }
    }
    // The next statement causes a compiler error.
    info(strfmt("Found: %1", i));
}

static void AnotherMethod()
{
    str textFromFile;
    using (System.IO.StreamReader sr = new System.IO.StreamReader("c:\test.txt"))
    {
        textFromFile = sr.ReadToEnd();
    }
}
```

You can also scope variables to a `using` statement, as shown in the following example.

```csharp
static void AnotherMethod()
{
    str textFromFile;
    using (System.IO.StreamReader sr = new System.IO.StreamReader("c:\test.txt"))
    {
        textFromFile = sr.ReadToEnd();
    }
}
```

When you use an object that implements `IDisposable`, you should declare and instantiate that object in a `using` statement. The `using` statement calls the `Dispose` method on the object correctly, even if an exception occurs while you’re calling methods on the object. You can achieve the same result by putting the object inside a `try` block and then explicitly calling `Dispose` in a `finally` block. In fact, the compiler translates the `using` statement in just this manner.

The following example shows some of the features that we have been describing.

```csharp
// loop variable declared within the loop: It will
// not be misused outside the loop
for(int i = 1; i < 10; i++)
{
    // Because this value is not used from outside the loop,
    // its declaration belongs in this smaller scope.
    str s = int2str(i);
    info(s);
}
```

To prevent confusion, the compiler issues an error message if you try to introduce a variable that will hide another variable in an enclosing scope, or even in the same scope. For example, the following code will cause the compiler to issue the following diagnostic message: "A local variable named 'i' can't be declared in this scope because it would give a different meaning to 'i', which is already used in a parent or current scope to denote something else."

```csharp
{
    int i;
    {
        int i;
    }
}
```

**Constants, read-only variables, and macros**
The concept of macros is fully supported. Constants have the following advantages over macros:

- You can add a documentation comment to a constant but not to the value of a macro. Eventually, the language service will pick up this comment and provide useful information to the user.
- A constant is known by IntelliSense.
- A constant is cross-referenced. Therefore, you can find all references for a specific constant but not for a macro.
- A constant is subject to access modifiers. You can use the `private`, `protected`, and `public` modifiers. The accessibility of macros isn’t rigorously defined.
- Constant variables have scope, whereas macros don’t have scope.
- You can see the value of a constant or a read-only variable in the debugger.

Macros that are defined in class scopes (that is, in class declarations) are effectively available in all methods of all derived classes. This feature was originally a bug in the implementation of the legacy compiler macro, however, many application programmers often take advantage of it now. The X++ compiler still honors this feature, but no new code that uses it should be written. This feature also has a significant effect on the performance of the compiler.

Constants can be declared at the class level, as shown in the following example.

```x++
private const str MyConstant = 'SomeValue';
```

The constants can then be referenced by using the double colon (:) syntax.

```x++
str value = MyClass::MyConstant;
```

If you’re in the scope of the class where the constant (`const`) is defined, you can omit the type name prefix (`MyClass` in the preceding example). Therefore, you can easily implement the concept of a macro library. The list of macro symbols becomes a class that has public `const` definitions.

You can also define constants as variables only. The compiler will maintain the invariant so that the value can’t be modified.

```x++
{
    const int Blue = 0x0000FF;
    const int Green = 0x00FF00;
    const int Red = 0xFF0000;
}
```

Read-only fields can only be assigned a value once, and that value never changes; the field can be assigned its value either inline, at the place where the field is declared, or in the constructor.

### Null values for data types

The concept of null values that is available in many other database management systems (DBMSs) is not supported. A variable in X++ always has a type and a value, however, for each data type, one value is considered null (for example, when the `validateField` table method is run).

<table>
<thead>
<tr>
<th>TYPE</th>
<th>VALUE THAT IS TREATED AS NULL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>1900-01-01</td>
</tr>
<tr>
<td>Enum</td>
<td>An element that has its value set to 0</td>
</tr>
<tr>
<td>TYPE</td>
<td>VALUE THAT IS TREATED AS NULL</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>Integer</td>
<td>0</td>
</tr>
<tr>
<td>Real</td>
<td>0.0</td>
</tr>
<tr>
<td>String</td>
<td>An empty string</td>
</tr>
<tr>
<td>Time</td>
<td>00:00:00</td>
</tr>
<tr>
<td>Utcdatetime</td>
<td>Any value that has its date portion set to 1900-01-01, regardless of the value of the time portion. For example, the value 1900-01-01T22:33:44 is treated as null. Note that any utcDateTime value that has its date portion set to 1900-01-01 is shown as blank by the X++ print statement. Only the value 1900-01-01T00:00:00 is shown as blank by the Global::info method. That value is the value from the DateTimeUtil::MinValue method.</td>
</tr>
</tbody>
</table>

When the `validateField` method checks whether a user has entered a value in a mandatory field, 0 isn’t accepted in an integer type field, the first entry isn’t accepted in an enum type field, and so on. Additionally, in SQL X++ statements, the values that are listed in the previous table yield false in a Boolean comparison. However, in non-SQL X++ statements, the equal and relational operators work with these values, just as they work with other values. Variables of the `container` type, and classes and variables of the `table` type can be null in the traditional DBMS sense. A `table` type is null if all its fields have their null value.

### Casting

Casting refers to assignments between variables whose declared types are both in the same inheritance chain. A cast is either a down-cast or an up-cast. Consider the following class hierarchy.

```
Animal
  Horse
  Mustang
MotorVehicle
    +void run()
```

The MotorVehicle class isn’t related to the Animal class, though both derive from Object. An **up-cast** happens when assigning an expression of a derived type to a base type:

```java
Animal a = new Horse();
```

A **down-cast** happens when assigning an expression of a base type to a derived variable.
Horse h = new Animal();

Both up-casts and down-casts are supported in X++. However, down-casts are dangerous and should be avoided whenever possible. The down-cast example above fails with an `InvalidCastException` at runtime, since the assignment doesn't make sense. X++ supports late binding on a few types, like `Object` and `FormRun`. This means that the compiler won't diagnose any errors at compile-time when it sees a method being called on those types, if that method isn't declared explicitly on the type. It's assumed that the developer knows what they're doing. For instance, the following code may be found in a form:

```csharp
Object o = element.args().caller();
o.MyMethod(3.14, "Banana");
```

The compiler can't check the parameters or return values for the `MyMethod` method, since this method isn't declared on the object class. At runtime, the call will be made using reflection, which is much slower than calls that don't require reflection. Calls to methods that are defined on the late-binding types will be checked. For example, the call to `ToString` in the following example:

```csharp
o.ToString(45);
```

will cause a compilation error:

```
'Object.toString' expects 0 argument(s), but 1 specified.
```

because the `ToString` method is defined on the `Object` class. There's one difference from the implementation in previous versions of X++ (AX2012 and earlier) related to the fact that methods could be called on unrelated objects, as long as the name of the method was correct, even if the parameter profiles weren't entirely correct. This is no longer supported.

**Example - casting**

```csharp
public class MyClass2
{
    public static void Main(Args a)
    {
        Object obj = new Car();
        Horse horse = obj; // exception now thrown
        horse.run(); // Used to call car.run()!
    }
}
```

You should use the `is` and `as` operators liberally in your code. The `is` operator can be used if the expression provided is of a particular type (including derived types). The `as` operator will perform casting into the given type and return null if a cast isn't possible.
This topic describes primitive data types in X++. The primitive data types in X++ are `anytype`, `boolean`, `date`, `enum`, `guid`, `int`, `int64`, `real`, `str`, `timeOfDay`, and `utcdatetime`.

**anytype**

The `anytype` data type is a placeholder for any data type. You should use variables of this type only as arguments and return values.

To use `anytype` as a variable, you must first assign a value to it, otherwise, a run-time error occurs. After you've assigned a value to `anytype`, you can't convert it to another data type.

Although you can use `anytype` variables in expressions, they're usually used as arguments and return types. The size, precision, scope, default value, and range of `anytype` depend on the conversion type that you assign to it. You can use `anytype` just as you use the data type that you convert it to. For example, if you assign an integer, you can then apply relational and arithmetic operators to the variable.

An `anytype` variable is automatically converted to a date, enumeration (enum), integer, real, string, extended data type (EDT) (record), class, or container when a value is assigned to the type. Additionally, the following explicit conversion functions can be used: `any2date`, `any2enum`, `any2int`, `any2real`, and `any2str`. You can't change the variable to another data type after you've converted it to `anytype`.

**anytype examples**

```csharp
// An example of using anytype variables.
public static str range(anytype _from, anytype _to)
{
    return queryValue(_from) + '..' + queryValue(_to);
}

// Another example of using anytype variables.
void put(int position, anytype data)
{
    record = conPoke (record, position, data);
}

public void AnytypeMethod()
{
    // An example of automatic conversion for anytype.
    anytype a;
    a = "text"; // Automatically assigns a string literal.
}
```

**boolean**

The `boolean` data type contains a value that is evaluated as either `true` or `false`. You can use the reserved literal keywords `true` and `false` wherever a `boolean` expression is expected. The size of a `boolean` is 1 byte. The default value is `false`, and the internal representation is a short number.

A `boolean` is automatically converted to an `int`, `date`, or `real`. It has no explicit conversion functions. The internal representation of a `boolean` is an integer. You can assign any integer value to a variable that is declared as the `boolean` type. The integer value 0 (zero) is evaluated as `false`, and all other integer values are evaluated
as true. Because the internal representation of a boolean is an integer, boolean values are automatically converted to integers and reals.

**boolean examples**

```java
public void BooleanMethod()
{
    // Simple declaration of a boolean variable, b.
    boolean b;

    // Multiple declarations of booleans.
    boolean b1, b2;

    // Boolean variable is initialized to true.
    boolean b3 = true;

    // Declares a dynamic array of booleans.
    boolean b4[];

    // This example shows the most common usage of a boolean: a boolean in
    // a conditional statement and as a result of a logical expression.
    void main()
    {
        // Declares a boolean called exprValue.
        boolean exprValue;

        // Assigns ExprValue the value of (7*6 == 42), which equates to true.
        exprValue = (7*6 == 42);

        // If the conditional statement is true, print "OK".
        if (exprValue)
        {
            print "OK";  //"OK" is printed because the expression is true.
        }
    }
}
```

**date**

The date data type contains the day, month, and year. Dates can be written as literals by using the following syntax: `Date literal = day \ month \ year`. You must use four digits for the year.

The date data type can hold dates between January 1, 1900, and December 31, 2154. The size of a date is 32-bits. The default value is null, and the internal representation is a date.

A date has no implicit conversions, however, the following explicit conversion functions can be used: `str2date`, `date2str`, `date2num`, and `int2date`.

You can add and subtract integers from dates, which moves the date some days into the future and past respectively. Subtracting dates from each other will calculate the difference in days, however, adding two dates together is not possible and will lead to a compiler error.

**date examples**
public void DateMethod()
{
    // Simple declaration of a date variable, d.
    date d;

    // Multiple declaration of two date variables.
    date d1, d2;

    // A date variable, d3, is initialized to the 21st of November 1998.
    date d3 = 21\11\1998;

    // Declaration of a dynamic array of dates.
    date d4[];

    // Using arithmetic operators with integer variables and dates.
    void myMethod()
    {
        int anInteger;
        date aDate;
        // Sets the date variable aDate to January 1, 1998.
        aDate = 1\1\1998;
        // Sets the integer variable anInteger to 30.
        anInteger = 30;
        // Uses an integer value in the computation of dates.
        // This sets aDate to aDate + 30; that is the 31st of January 1998.
        aDate = aDate + anInteger;

        // Create 2 variables, set bDate, and then subtract from that date.
        date bDate;
        int dateDifference;
        bDate = 2\10\1998;
        dateDifference = bDate - aDate; // dateDifference will equal 244.
    }
}

enum
An enum is a list of literals. Before you can use an enum, you must declare it in Application Explorer.

The literal values are represented internally as integers. The first literal has the number 0, the next literal has the number 1, the next literal has the number 2, and so on. You can use enum values as integers in expressions. The default value for the first entry is 0, and the internal representation is a short number.

An enum value is automatically converted to a boolean, int, or real. Additionally, the following explicit conversion functions can be used: enum2str and str2enum.

Hundreds of enumerable types are built into the standard application. For example, the NoYes enum has two associated literals: No has the value 0, and Yes has the value 1. You can create as many enum types as you want, and you can declare up to 251 (0 to 250) literals in a single enum type. To reference an enum value, enter the name of the enum, two colons, and then the name of the literal. For example, to use the literal No in the NoYes enum, enter NoYes::No.

Create an enum
1. In Solution Explorer, right-click the project, point to Add, and then click New Item.
2. Under Artifacts, select Data Types.
3. Click Base Enum to select the new item type.
4. In the Name field, enter a name for the enum, and then click Add. A new enum is added to the project, and the enum designer for the new element is opened.
5. In the enum designer, right-click the enum name, and then click New Element.
6. In the Properties window, enter the name of the enum element.
enum examples

```java
public void EnumMethod()
{
    // Declare the enum (a NoYes enum) in the Application Explorer.
    NoYes done;

    // An array of Criteria enums.
    Criteria crit[100];
}
```

guid

The `guid` data type holds a **globally unique identifier** (GUID) value. A GUID is an integer that can be used across all computers and networks, wherever a unique identifier is required. It's unlikely that the number will be duplicated. A valid GUID meets all the following specifications:

- It must have 32 hexadecimal digits.
- It must have four dash characters that are embedded at the following locations: 8-4-4-4-12.
- Braces ({} at the beginning and end of a string are optional. For example, both "12345678-BBBb-cCCCC-0000-123456789012" and "(12345678-BBBb-cCCCC-0000-123456789012)" are valid GUID strings.
- It must have a total of either 36 or 38 characters, depending on whether braces are added.
- The hexadecimal digits a–f (or A–F) can be uppercase, lowercase, or mixed.

The size of a `guid` is 16 bytes or 128-bits. The following explicit conversion functions can be used: `any2guid`, `guid2str`, `newGuid`, `str2guid`, `Global::guidFromString`, and `Global::stringFromGuid`.

guid examples

The following set of examples shows how to use the `guid` functions. The code output of these examples follows.

```java
// An example of how to use the GUID functions.
static void GuidRoundTripJob(Args _args)
{
    guid guid2;
    str string3;

    // Convert a guid to a string, and back to a guid.
    guid2 = newGuid();
    info(strFmt("Info_a1:  guid2 == %1", guid2));
    string3 = guid2str(guid2);
    info(strFmt("Info_a2:  string3 == %1", string3));
    guid2 = str2guid(string3);
    info(strFmt("Info_a3:  guid2 == %1", guid2));

    // Test string representations of a guid. Mixing upper and lower case letters does not affect the guid.
    guid2 = str2guid("BB345678-abcd-ABCD-0000-bbbbffff9012");
    string3 = guid2str(guid2);
    info(strFmt("Info_b1:  8-4-4-4-12 format for dashes works (%1)", string3));
    info(strFmt("Info_b2:  Mixed upper and lower case works."));

    // Test invalid dash locations, see output is all zeros. Dash locations must be exact.
    guid2 = str2guid("CC2345678abcd-ABCD-0000-cccc9012");
    string3 = guid2str(guid2);
    info(strFmt("Info_c1:  These embedded dash locations are required. %1", string3));

    // Braces {} are optional.
    guid2 = str2guid("DD345678-abcd-ABCD-0000-ddddaaaa9012");
    string3 = guid2str(guid2);
    info(strFmt("Info_d1:  Braces {} are optional (%1)", string3));
}
```
int and int64

Integers are numbers that have no decimal places. There are two integer types: int and int64. Integers are used as control variables in repetitive statements or as indexes in arrays.

You can also use integer literals anywhere that an integer expression is expected, and both relational and arithmetic operators can be applied. An integer literal is the integer as it's entered directly in the code, such as 32768. An int is 32-bits wide, and an int64 is 64-bits wide. The default value is 0, and the internal representation is a long number. An integer is automatically converted to a boolean, enum, or real.

Additionally, the following explicit conversion functions can be used: str2int, int2str, str2int64, and int642str. The range of an int is [-2,147,483,647 : 2,147,483,647], and the range of an int64 is [-9,223,372,036,854,775,808 : 9,223,372,036,854,775,808]. All integers in either of these ranges can be used as literals.

int and int64 examples

The following example shows how to declare integers and use them in expressions. If you try to assign the largest integer plus 1 to an int64, you get the wrong result, because the number is interpreted as a 32-bit number. Therefore, the number is wrapped around and stored instead as -2,147,483,647. To prevent this behavior, add "u" to the end of the number. For example, enter int64 i = 0x8000 0000u (0x8000 0000 is 2,147,483,648).
public void IntegerMethod()
{
// Declaration of an integer variable, i.
    int i;

// Declaration of two int64 variables.
    int64 i1, i2;

// An integer variable is initialized to the value 100.
    int i3 = 100;

// Declaration of a dynamic array of integers.
    int i4[];

    void element()
    {
        // Two integer variables are declared and initialized.
        int k = 1, j = 2;

        // j is assigned the result of j + ((i + i) DIV 2).
        j += (i + i) div 2;

        // This results in: j=3.
        if (j > 2 )
        {
            print "J is greater than 2";
        }
        else
        {
            print "J is NOT greater than 2";
        }
    }
}

real

A real variable can hold decimal values in addition to integers. You can use decimal literals anywhere that a real is expected. A decimal literal is the decimal as it’s entered directly in the code, such as 2.123876. Real literals can also be written by using exponential notation, such as 1.0e3.

Reals can be used in all expressions, and they can be used with both relational and arithmetic operators. A real has a precision of 16 significant digits. The default value for a real is 0.0, and the internal representation is a binary-coded digital (BCD) number. The BCD encoding enables exact representations of values that are multiples of 0.1. The range of a real variable is -(10)¹²⁷ through (10)¹²⁷. All reals in this range can be used as literals in X++.

A real variable is automatically converted to a boolean, enum, or int. If the result is an integer, or if the operator is an integer operator, the real is converted to an integer. If the result is a boolean, the real is converted to a boolean, and so on. Additionally, the following explicit conversion functions can be used: str2num and num2str.

Direct assignments between X++ real and the Microsoft .NET Framework System.Decimal convert the value correctly. A call to a conversion function isn’t required. A decimal number is a floating-point value that consists of a sign, a numeric value where each digit is in the range 0 through 9, and a scaling factor that indicates the position of a floating decimal point that separates the integral and fractional parts of the numeric value. The binary representation of a real value consists of a 1-bit sign, a 96-bit integer number, and a scaling factor. The scaling factor is used to divide the 96-bit integer and specify what part of it is a decimal fraction. The scaling factor is implicitly the number 10 raised to an exponent in the range 0 through 28. Therefore, the binary representation of a decimal value represents ([-2⁹⁶ through 2⁹⁶] ÷ 10(0 \ through \ 28)), where -(2⁹⁶-1) is the minimum value that can be expressed and 2⁹⁶-1 is the maximum value.
NOTE
The type that is used to represent real values in Finance and Operations applications has changed from the interpreted X++ of Microsoft Dynamics AX 2012. However, you don’t have to rewrite any code, because the new type can express all the values that the old type could express. We provide this material in the interest of full disclosure only.

All instances of the real type are now implemented as instances of the .NET decimal type (System.Decimal). Just as the real type in previous versions, the decimal type in a binary-coded decimal type is resilient to rounding errors. The range and resolution of the decimal type differ from previous versions. The original X++ real type supported 16 digits and an exponent that defined the position of the decimal point. However, the X++ real type for Finance and Operations applications can represent decimal numbers in the range 79,228,162,514,264,337,593,543,950,335 (2^96-1) through -79,228,162,514,264,337,593,543,950,335 (-[2^96-1]).

Rounding is still required for the new real type. For example, the following code produces a result of 0.9999999999999999999999999999 instead of 1. No number of decimals will suffice to represent the value of 1/3 accurately. The discrepancy obtained here is due to the fact that only a finite number of decimals are provided. You should use the round function to round to the number of decimals required.

```csharp
// An example of using the debugger to show the value of the variables.
public static void UseTheDebugger(Args a)
{
    real dividend = 1.0;
    real divisor = 3.0;
    str stringvalue;
    System.Decimal valueAsDecimal;
    real value = dividend/divisor * divisor;
    valueAsDecimal = value;
    info(valueAsDecimal.ToString("G28"));
    // An example of using the Round function to round to the number of decimals required.
    value = round(value, 2);
}
```

real examples

```csharp
public void RealMethod()
{
    // Simple declaration of a real variable, r.
    real r;

    // Multiple declaration of two real variables.
    real r1, r2;

    // A real variable is initialized to the approximate value of pi.
    real r3 = 3.1415;

    // Declaration of a dynamic array of reals.
    real r4[];

    // An example of a real literal written using exponential notation.
    real r;
    r = 1.000e3;
    r = 1.2345e+3;
    r = 1.2345e+03;
    r = 1234.5e4;
    r = 1.0e1; // Means 1.0E1
}
```

// An example of automatic conversions.
void main()
{
    // Declares a variable of type integer with the name exprValue.
int exprValue;

    // Declares a real variable with the name area.
real area = 3.141528;
exprValue = Area/3;

    // The expression Area/3 is a real expression because
    // division is a real operator, and the result is 1.047176. This result is
    // automatically converted (actually truncated) to an integer with the value 1,
    // because exprValue is an integer.
}

    // An example of a real being converted to .NET System.Decimal.
void AnotherMain(Args _args)
{
    real real9;
    System.Decimal sysdec1;

    // Direct assignments supported between these types.
    sysdec1 = 2.3456;
    real9 = sysdec1;
    info(strFmt("strFmt says real9 == %1", real9));
}

    /***
Message (05:48:43 pm)
strFmt says real9 == 2.35
***/

    // An example of using reals in expressions.
void myMethod()
{
    // Two real variables are declared and initialized.
    real i = 2.5, j = 2.5;

    // j is assigned the result of j * i, so j=6.25.
    j = j * i;
    if (j > (i * 2)) // If j > 5
    {
        print "Great"; // "Great" is printed.
    }
    else
    {
        print "Oops"; // else "Oops" is printed.
    }
}

str

A str variable (a string) is a sequence of characters that are used as texts, help lines, addresses, telephone numbers, and so on.

To declare a string, use the str keyword.

String literals are characters that are enclosed in quotation marks (""). String literals can be used wherever string expressions are expected. Examples of string literals include "StringLit" and "Hello World". If you want the string to span more than one line, precede it with an at sign (@). You can use strings in logical expressions, such as comparisons. You can also concatenate strings by using the + operator.

The default value for a string is empty, and the internal representation is a list of characters. There are no automatic conversions for strings, however, the following explicit conversion functions can be used: str2int, str2int64, int2str, str2num, num2str, str2date, and date2str.

A string can hold an unlimited number of characters, however, you can specify the maximum length of a string
in the variable declaration. The string is then truncated to that maximum length. An example is shown in the next section.

**str examples**

```c++
void StringMethod()
{
    // Declare a dynamic string of unlimited length.
    str unlimitedString;

    // Declare a string with a maximum of 64 characters
    // in order to force a truncation, initialized to "A".
    str 64 maxLengthString = "A";

    // Declare an array of 100 strings.
    str 30 hundredStrings[100];

    // Using strings in expressions.
    void myMethod()
    {
        // Two strings are declared and initialized.
        str a="Hello", b="World";

        // The concatenation of a, " " and b is printed in a window.
        print a+" "+b;
    }
}
```

**String truncation**

String values can be declared in X++ to contain a maximum number of characters. Typically, this is achieved by encoding this information in an extended data type and setting the **String Size** in the **Properties** window. In the following screenshot, **FMCreditCardNum** cannot exceed 20 characters.
Run the following code in the debugger by including it in a static `Main` method to observed the behavior.

```java
creditCardNumber = "12345678901234567890Excess string";
```

You can also specify length constraints in X++:

```java
str 20 creditCardNumber;
```

All assignments to these values are implicitly truncated to this maximum length.

**timeOfDay**

The `timeOfDay` (time) data type is an integer value that represents the number of seconds that have passed since midnight. Like integers, `timeOfDay` variables can be used as literals. Relational and arithmetic operators can be applied to `timeOfDay` variables. A `timeOfDay` variable can also be used in expressions. The range of a `timeOfDay` data type is in the closed interval $[0; 86,400]$. Values above 86,400 (23:59:59) can't be interpreted. A `timeOfDay` variable is automatically converted to a `boolean`, `enum`, or `real`. Additionally, the following explicit conversion functions can be used: `str2time` and `time2str`.

**timeOfDay examples**
public void TimeofdayMethod()
{
    // Declaration of a timeOfDay variable, time1.
    timeOfDay time1;

    // Declaration and initialization of a timeOfDay variable to 00:21:35.
    timeOfDay time2 = 1295;
}

utcdatetime

The utcdatetime data type combines the date type and the timeOfDay type. A utcdatetime variable also holds information about the time zone, however, this information can’t be accessed in code.

The format for a utcdatetime literal is yyyy-mm-ddThh:mm:ss. The uppercase "T" is required. This format can be written without quotation marks. The minimum value is 1900-01-01T00:00:00, and the maximum value is 2154-12-31T23:59:59. This maximum value matches the upper range of date and timeOfDay. The smallest unit of time in utcdatetime is one second.

A utcdatetime variable that has been declared but hasn’t been initialized has the default value 1900-01-01T00:00:00. This value is the value that is returned by DateTimeUtil::minValue(). Some functions treat an input parameter of this minimum value as null. For example, the DateTimeUtil::toStr method returns an empty string, however, the DateTimeUtil::addSeconds method returns a usable utcdatetime value.

There are no implicit conversions for the utcdatetime data type. The DateTimeUtil class provides many methods that you can use to manipulate utcdatetime values.

The following explicit conversion functions can also be used: str2datetime and datetime2str.

Additionally, the Global class provides the utcDateTime2SystemDateTime and CLRSystemDateTime2UtcDateTime conversion methods to support common language runtime (CLR) interop.

Comparison operators are the only type of operators that can be used with the utcdatetime data type. The following operators can be used to compare two utcdatetime values: !=, <, <=, ==, >, and >=. When you add a utcdatetime field to a table, we recommend that you base the field on an EDT.

utcdatetime examples

public void UtcdatetimeMethod()
{
    // Declaring a utcdatetime literal.
    utcdatetime myUtc2 = 1988-07-20T13:34:45;

    // Another example of declaring a utcdatetime literal.
    int iDay = DateTimeUtil::day(1988-07-20T13:34:45);

    // utcdatetime using a quoted string parameter into the DateTimeUtil::parse method.
    utcdatetime myUtc4 = DateTimeUtil::parse("1988-07-20T13:34:45");
}
This topic describes composite data types in X++. The composite data types in X++ are arrays, containers, classes as data types, delegates as data types, and tables as data types.

Array

An array is a variable that contains a list of items that have the same data type. The elements of an array are accessed by using integer indexes. You use a separate statement to initialize each element in an array. When you use a container data type or an array object to create a collection, you can initialize multiple elements by using a single statement. By default, all the items in an array have the default value of the data type in the array. There are three kinds of arrays: dynamic arrays, fixed-length arrays, and partly on disk arrays.

- **Dynamic arrays** – These arrays are declared by using an empty array option. In other words, they have only brackets ([]).
- **Fixed-length arrays** – These arrays can hold the number of items that is specified in the declaration. Fixed-length arrays are declared like dynamic arrays, but a length option is included in the brackets.
- **Partly on disk arrays** – These arrays are declared as either dynamic arrays or fixed-length arrays that have an extra option that declares how many items should be held in memory. The other items are stored on disk and are automatically loaded when they are referenced.

X++ supports only one-dimensional arrays. However, you can mimic the behavior of multiple array indexes. (For more information, see the Multiple array indexes section). Variables in objects and tables can be declared as arrays. For example, this functionality is used in address lines in the standard application. An array collection class lets you store objects in an array.

Array indexes begin at 1. The first item in the array is referenced as [1], the second item is referenced as [2], and so on. The following syntax is used to access an array element: \[\text{ArrayItemReference} = \text{ArrayVariable}\[\text{Index}\}\]. In this syntax, \text{ArrayVariable}\ is the identifier of the array, and \text{Index}\ is the number of the array element. \text{Index} can be an integer expression. Item zero [0] is used to clear the array. If a value is assigned to index 0 in an array, all elements in the array are reset to their default value.

An assignment of one entire array to another is performed by reference.

*Array examples*
public void ArrayMethod()
{
    int myArray[10]; // Fixed-length array with 10 integers.
    myArray[0] = 0; // Resets all elements in intArray.

    // Dynamic array of integers.
    int intArray[];

    // Dynamic array of variables of type Datatype.
    //Datatype arrayVariable[];

    // Fixed-length arrays.
    boolean boolArray[100]; // Fixed-length array of booleans with 100 items.

    // Two examples of Partly On Disk Arrays.
    // Dynamic integer array with only 100 elements in memory.
    int arrayVariable[ ,100];
    // Fixed-length string array with 1000 elements, and only 200 in memory.
    str arrayVariable2[1000,200];

    // A dynamic array of integers.
    int i[];
    // A fixed-length real array with 100 elements.
    real r[100];
    // A dynamic array of dates with only 10 elements in memory.
    date d[ ,10];
    // A fixed length array of NoYes variables with 100 elements and 10 in memory.
    NoYes ny[100,10];
}

Multiple array indexes

Some languages, such as C++ and C#, let you declare arrays that have more than one index. In other words, you can define "arrays of arrays." In X++, you can’t directly create multiple array indexes because only one-dimensional arrays are supported. However, you can implement multiple indexes by using the method that is described in this section. For example, you want to declare an array that has two dimensions, to hold an amount that is earned by country by dimension. There are 10 countries and three dimensions. In C++ and C#, you declare the following array.

    // This is C# or C++ code, not X++ code.
    real earning[10, 3];

However, X++ doesn’t support this declaration. Instead, you can define a one-dimensional array where the number of elements is the product of the elements in each dimension. Here is an example.
public void MultipleArrayMethod()
{
    // Step 1: define a one-dimensional array with the number
    // of elements that is the product of the elements in each dimension.
    real earnings[10*3];

    // Step 2: to refer to a specific element, such as earnings[i,j], write the following:
    // declare i and j (maybe) and assign the value to something
    int i = 1;
    int j = 2;
    real element = earnings[(i-1)*3 + j];
}

// This can be written into a macro like this:
#localmacro.earningIndex
(%1-1)*3+%2
#endmacro

public void CallTheMacro()
{
    // Next, call the specific element within the macro like this:
    int i = 1;
    int j = 2;
    real element = earnings[earningIndex(i,j)];

    // The previous scheme can be extended to any number of dimensions.
    // The element a[i1, i2, ..., ik] can be accessed by computing the
    // offset into an array containing (d1*d2*...*dk) elements.
    //((i1 - 1)*d2*d3*...*dk +
    //((i2 - 1)*d3*d4*...*dk + .... +
    //((ik-1 -1)*dk +
    //((ik-1)


Container

A container object is a dynamic list of items that contains primitive data types or composite data types. A container is useful when you must pass various types of values between the client and server tiers. However, if you plan to repeatedly add to a list in a loop, a container isn't a good choice. Containers are most suitable for processes that don't involve excessive modification to the size or contents of the container. When a container undergoes excessive additions of data, overall system performance can decrease because container data must be repeatedly copied and new space must be repeatedly allocated.

A container isn't a class. A container contains an ordered sequence of primitive values or other containers. Because of the flexibility of anytype, a container offers a good way to store values of different types together. A container can be stored in the database. A container is one of the column types that you can select when you use Application Explorer to add a column to a table. A container slightly resembles an array, or collections, such as the List or Stack classes. However, you can never change the size or content of a container after the container is created.

X++ statements that appear to modify a container are internally building a new container and copying values as required. Even an assignment of a container to another container variable creates a new copy of the container. All these operations can affect performance. In the functions that provide access to a container (such as conPeek), the container is 1-based, not 0-based. Indexing is 1-based for arrays. The default value of a container is empty. The container doesn't contain any values. Some statements that use containers might appear to modify a container, however, inside the system, containers are never modified. Instead, the data from the original container is combined with data from the command to build a new container. You can create a new container by using one of the following functions: conDel, conIns, or conPoke.

Additionally, the Global class has static methods for handling containers. These methods include
Comparing container to other options

The container type resembles other constructs, such as arrays and collection classes, such as List and Stack. The difference between a container and List is that an instance of the List class is mutable. A List object first allocates more space than its data consumes. Then, as data is added, the space is filled. This behavior is more efficient than allocating more space every time that an element is added. An update of a List performs faster than similar operations on a container.

When you construct a List object, you determine the one type of data that the List object can store. This restriction is less flexible for a List than it is for a container. However, you can store objects in a List, whereas a container can store only value types. The difference between a container and an array is that an array can hold only items of its declared type. You can allocate memory space for an array and fill that space with values later. For example, you can fill in values in a loop. This behavior is efficient and performs well. When you want to build a new container by appending new data, you can use either the += operator or the conIns function. The += operator is the faster alternative. Use the conIns function only when you want to add new data before the last index of the original data.

You can’t store object references in containers. When the compiler detects an attempt to store an object reference in a container, it issues an error message. If the type of the element that is added to the container is anytype, the compiler can’t determine whether the value is a reference type. In this case, the compiler allows the attempt. Although the compiler doesn’t diagnose the code as erroneous, an error will be thrown at run time.

Container examples

```java
public void ContainerExample()
{
    // First, declare the variables you are using.
    container myContainer;
    container myContainer4;
    container myContainer5;
    // Three ways to declare a container.
    myContainer = [1];
    myContainer += [2];
    myContainer4 = myContainer5;

    // Declare a container.
    container cr3;

    // Assign a literal container to a container variable.
    cr3 = [22, "blue"];

    // Declare and assign a container.
    container cr2 = [1, "blue", true];

    // Mimic container modification (implicitly creates a copy).
    cr3 += [16, strMyColorString];
    cr3 = conIns(cr3, 1, 3.14);
    cr3 = conPoke(cr3, 2, "violet");

    // Assignment of a container (implicitly creates a copy).
    cr2 = cr3;

    // Read a value from the container.
```
str myStr = conPeek(cr2, 1);

// One statement that does multiple assignments from a container.
str myStr;
int myInt;
container cr4 = ["Hello", 22, 2007, 1988];
[myStr, myInt] = cr4; // "Hello", 22

// Example of applying the = operator to a container. The example
// initializes myContainer2 and myContainer33.
myContainer2 = [2, "apple"];

// Next, you make a copy of myContainer33 and assign the copy to myContainer2.
myContainer33 = [33, "grape"]; myContainer2 = myContainer33; // The container that myContainer2 had been holding is no longer
available and cannot be recovered.

// An example of building a new container by
// assigning a new value to myContainer33 through the += operator.
myContainer33 += [34, "banana"];
}

// Container example. In this example, variable2 and variable33 hold different containers.
static void JobC(Args _args)
{
    container variable2, variable33;
    variable2 += [98];
    variable33 = variable2;
    variable2 += [97];
}

// List class example. In this example, variable2 and variable33 refer to the same List object.
static void JobL(Args _args)
{
    List variable2, variable33;
    variable2 = new List(Types::Integer);
    variable2.addEnd(98);
    variable33 = variable2;
    variable2.addEnd(97);
}

// The automatic type conversion by anytype also applies to the special syntax for making multiple
// assignments from a container in one statement. This is shown in the following code example,
// which assigns a str to an int, and an int to a str.
static void JobContainerMultiAssignmentUsesAnytype(Args _args)
{
    container con2;
    int int4;
    str str7;
    con2 = ["11", 222];
    [int4, str7] = con2;
    info(strfmt("int4==11==(%1), str7==222==(%2)", int4, str7));
}

/*** Output:
Message (10:36:22 am)
int4==11==(11), str7==222==(222)
****/

static void UseQuery()
{
    // An example of how the compiler diagnoses attempts to store object in containers
    container c = [new Query()]; // This statement will cause the error message shown below.
    /*** Instance of type 'Query' cannot be added to a container. ***/

    // An example of a code that won't cause an error message, but will
    // cause an error message to be thrown at runtime.
    anytype a = new Query();
    container d = [a];
Classes as data types

A class is a type definition that describes both variables and methods for instances of the class. (The instances of a class are also known as objects.) A class is only a definition for objects, and all objects are null when they are declared. In Application Explorer, every application class under the Classes node is a data type. You can declare variables of these types in your code. You can construct instances of a class and assign the instances to variables.

Classes can be nested in source code. Nested classes are available only inside forms (such as a class that extends FormRun), and are used to represent controls, data sources, or data fields. An attribute decoration, such as the attribute decoration on a class or a method, can omit the suffix of the attribute name if the suffix is Attribute. Therefore, X++ allows [MyFavorite] instead of requiring [MyFavoriteAttribute]. Additionally, attributes are now applied to the handlers of delegates and methods, to map the handlers to those targets.

In AX 2012 and earlier versions, you could designate a method to run on either the client or the server. However, in Finance and Operations applications, all compiled X++ code is run as .NET Common Intermediate Language (CIL) on the server. There is no longer any code that is evaluated at the client site or in the browser. Therefore, the client and server keywords are now ignored. Although these keywords don’t cause a compile error if they are used, they should not be used in any new code.

Private and protected member variables

Previously, all member variables that were defined in a class were protected. You can now make the visibility of member variables explicit by adding the private, protected, and public keywords. The interpretation of these modifiers is obvious and is aligned with the semantics for methods:

- **private** – The member variable can be used only within the class where it’s defined.
- **protected** – The member variable can be used in the class where it’s defined and all subclasses of that class.
- **public** – The member variable can be used anywhere. It’s visible outside the confines of the class hierarchy where it’s defined.

By default, member variables that aren’t adorned with an explicit modifier are still protected. However, as a best practice, you should explicitly specify the visibility. As we described earlier, when a member variable is defined as public, it can be accessed outside the class where it’s defined. In this case, you must specify a qualifier that designates the object that is hosting the variable. To specify the qualifier, use the dot notation, as you do for method calls.

In the following example, field1 is accessed by using the explicit this qualifier. In this case, it might not be a good idea to make a member variable public, because that approach exposes the internal workings of the class to its consumers, and therefore creates a strong dependency between the class implementation and its consumers. You should always try to depend only on a contract, not an implementation.

```plaintext
public class AnotherClass3
{
    int field1;
    str field2;
    void new()
    {
        this.field1 = 1;  // Explicit object designated.
        field2 = "Banana";  // 'this' assumed, as usual.
    }
}
```

Static constructors and static fields

Static fields are fields that are declared by using the static keyword. Conceptually, static fields apply to the class, not to instances of the class. Static constructors are guaranteed to run before any static calls or instance calls are
made to the class. The execution of the static constructor is relative to the user’s session. You never call the static constructor explicitly. Instead, the compiler will generate code to make sure that the constructor is called exactly one time, before any other method on the class is called. A static constructor is used to initialize any static data or perform an action that must be performed only one time. You can’t provide parameters for the static constructor, and it must be marked with the `static` keyword.

```csharp
// An example of how a singleton (call instance in the example below)
// can be created using the static constructor.
public class Singleton
{
    private static Singleton instance;
    private void new()
    {
    }
    static void TypeNew() // This is the static constructor.
    {
        instance = new Singleton();
    }
    public static Singleton Instance()
    {
        return Singleton::instance;
    }
}
// The singleton ensures that only one instance of the class
// will be called, which is consumed by the following.
{
    // Your code here.
    Singleton i = Singleton::Instance();
}
```

**Class elements in Application Explorer**

Under most class nodes in Application Explorer, there are two special nodes: a `classDeclaration` node and a `new` node. A `classDeclaration` always contains the X++ `class` keyword. Additional keywords, such as `extends`, can be included to modify the class. This node can also contain declarations of member variables.

In the following example, the variables `m_priority` and `m_rectangle` are members of the class.

```csharp
// An example of a classDeclaration.
public class YourDerivedClass extends YourBaseClass
{
    int m_priority;
    Rectangle m_rectangle;
    void new(int _length, int _width)
    {
        this.m_rectangle = new Rectangle(_length, _width);
    }
}
```

A `new` operator contains logic that is run when the `new` operator is used to create an instance of the class. The logic in the `new` method might construct an object and assign that object to a variable that is declared in the `classDeclaration`. Each class can have only one `new` method. However, in the `new` method, you often should call the `new` method of the base class. To call the `new` method of the base class, call `super()`.

The following example shows the `new` method for the `YourDerivedClass` class in the previous `classDeclaration` example. In this `new` method, the code constructs an instance of the `Rectangle` class. The instance is assigned to the `m_rectangle` variable. The `this` keyword that is used in the example is optional, however, if you include it, IntelliSense might be more helpful.
// An example of the new method from the previous classDeclaration example.
void new(int _length, int _width)
{
    this.m_rectangle = new Rectangle(_length, _width);
}

Garbage collection
Eventually during run time, most objects no longer have any variable that points to them. The system scans for
these objects and erases them from memory. The memory space then becomes available for other uses. The
Object class has a method that is named finalize. However, the finalize method isn't a destructor. The runtime
never calls the finalize method, even when an object is collected as garbage.

System classes
In Application Explorer, under System Documentation > Classes, there is a list of the kernel classes or system
classes. System classes aren’t written in X++, and you can’t see their source code. You can’t add system classes.
System classes usually have a new method, but they don’t have a classDeclaration node. Every application
class implicitly extends the Object system class. Some system classes are extended by an application class that
has a similar name. For instance, xClassFactory is extended by ClassFactory. In these cases, you should not
use the system class. For more information, see "Substitute application classes for system classes" in Classes
and methods.

Extension methods
The extension method feature lets you add extension methods to a target class by writing the methods in a
separate extension class. The following rules apply:

- The extension class must be static.
- The name of the extension class must end with the ten-character suffix _Extension, however, there’s no
  restriction on the part of the name that precedes the suffix.
- Every extension method in the extension class must be declared as public static.
- The first parameter in every extension method is the type that the extension method extends. However, when
  the extension method is called, the caller must not pass in anything for the first parameter. Instead, the
  system automatically passes in the required object for the first parameter.
- The target of an extension method must be a class, table, view, or map application object type.

An extension class can contain private or protected static methods. These methods are typically used for
implementation details and aren’t exposed as extensions. The extension method technique doesn’t affect the
source code of the class that it extends, therefore, the addition to the class doesn’t require over-layering.

Upgrades to the target class are never affected by any existing extension methods. If an upgrade to the target
class adds a method that has the same name as your extension method, your extension method can no longer
be reached through objects of the target class. The extension method technique uses the same dot-delimited
syntax that you often use to call regular instance methods. Extension methods can access all public artifacts of
the target class, but they can’t access anything that is protected or private. Therefore, extension methods can be
considered a type of syntactic sugar. Regardless of the target type, an extension class is used to add extension
methods to the type. For example, an extension table isn't used to add methods to a table, and there's no such
thing as an extension table.
public static class AtlInventLocation_Extension
{
    public static InventLocation refillEnabled(InventLocation _warehouse, boolean _isRefillEnabled = true)
    {
        _warehouse.ReqRefill = _isRefillEnabled;
        return _warehouse;
    }

    public static InventLocation save(InventLocation _warehouse)
    {
        _warehouse.write();
        return _warehouse;
    }
}

Delegates as data types

A delegate collects methods that subscribe to it. The delegate specifies the parameter signature that all its subscriber methods must share. When the delegate is called, the delegate calls each of its subscribers. A delegate never returns a value and can't have a default value. At first, every delegate has no subscribed methods. There is no limit on the number of parameters that a delegate can declare, and there is no limitation on the type of those parameters. The delegate body is always empty, because the delegate's only purpose is to define the contract that subscribers must conform to. A delegate doesn't have to be defined in a class. Delegates can also be defined in a table, form, or query.

Delegate examples

abstract class VarDatClass
{
    // delegatemethod examples
    // An example of declaring a delegate.
    delegate void notifyChange(utcdatetime _dateTime, str _changeDescription)
    {
    }

    // An example of subscribing an event handler to a delegate.
    public static void notifyStatic(utcDateTime _dateTime, str _changeDescription)
    {
        info("A notification has occurred calling static handler:" +
            DateTimeUtil::toStr(_dateTime) +
            " Message:" +
            _changeDescription);
    }
}

Tables as data types

All tables can be treated as class definitions. A table variable can be considered an instance (object) of the table (class) definition. For every field in a table variable, the default value is empty. You can address fields and create methods on tables. The methods can be invoked on instances of the table. To manipulate (that is, read, update, insert, and delete) records in tables, you must declare at least one table variable that can hold the record in focus. As a best practice, you should use the name of the table as the name of the variable but use an initial lowercase letter. Here are a few important differences between tables and objects:

- You can't allocate space for table variables. Allocation is done implicitly.
- Fields in table variables are public. You can reference them anywhere.
Fields in table variables can be referenced by using expressions.

There is no automatic conversion, but table variables that are declared as **Common** can hold data from any table.

**Scope of table variables**

In most respects, table variables can be considered objects, however, unlike objects, they aren't explicitly allocated. Only a variable declaration is required. All tables are compatible with the **Common** table, just as all objects are compatible with the **Object** class. Table variables are declared as common buffers and can be used to hold data from any table. You can't access tables that don't have table variables. The principles for declaring table variables and objects are the same, except with regard to the allocation of space.

**Table examples**

The syntax enables various possibilities for referencing fields in records. For example, you can use the `TableName.(FieldId)` syntax.

The following example prints the contents of the fields in the current record in the Customer table.

```java
// Declares and allocates space for one CustTable record.
public void myMethod()
{
    CustomerTable custTable;
}

// An example of referencing table variables.
public void printAccountNo()
{
    CustomerTable custTable;
    print custTable.AccountNo;  // Prints the field reference.
}
```

The following example uses the `fieldCnt` and `fieldCnt2Id` methods. The `fieldCnt` method counts the number of fields in a table, whereas `fieldCnt2Id` returns the ID for a field number. For example, you can use the `fieldCnt2Id` method to learn that field number 6 in a table has the ID 54. This conversion is required, because there is no guarantee that the IDs of the fields in a table are consecutive.

```java
// An example of the various possibilities for referencing fields in records.
public void printCust()
{
    int i, n, k;
    CustomerTable custTable;
    DictTable dictTable;
    dictTable = new DictTable(custTable.TableId);
    n = dictTable.fieldCnt();
    print "Number of fields in table: ", n;
    for (i=1; i<=n; i++)
    {
        k = dictTable.fieldCnt2Id(i);
        print "The ", dictTable.fieldName(k),
            " field with Id=" , k , " contains ",
            custTable.(k), ";";
    }
}
```
The X++ language syntax provides two composite types: arrays and containers. These composite types are useful for aggregating values of primitive types. However, you can't store class objects in arrays or containers.

**Collection classes** are used to store objects. They let you create arrays, lists, sets, maps, and structs that can hold any data type, even objects. For maximum performance, the classes are implemented in C++ (they are system classes). Collection classes were previously known as **foundation classes**. The collection classes are **Array**, **List**, **Map**, **Set**, and **Struct**.

- **Array** – This class resembles the **array** type in the X++ language, but it can hold values of any single type, even objects and records. Objects are accessed in a specific order.

- **List** – This class contains elements that are accessed sequentially. Unlike an array, the **List** class provides an **addStart** method. Like the **Set** class, the **List** class provides the **getEnumerator** and **getIterator** methods. You can use an iterator to insert and delete items from a **List** object.

- **Map** – This class associates a key value with another value.

- **Set** – This class holds values of any single type. Values aren't stored in the sequence in which they are added. Instead, the **Set** object stores the value in a manner that optimizes performance for the **in** method. A **Set** object ignores any attempt to add a value that the **Set** object is already storing. Unlike the **Array** class, the **Set** class provides the **in** and **remove** methods.

- **Struct** – This class can contain values of more than one type. It's used to group information about a specific entity.

The constructor for every collection class except **Struct** takes a type parameter that is an element of the **Types** system enum. The collection instance can store items of that type only. The **Types::AnyType** enum element is a special case that can't be used to construct a collection object, such as a **Set** object. The **null** value can't be stored as an element in a **Set** object. Additionally, **null** can't be a key in a **Map** object. You can iterate through a collection object by using an iterator or enumerator. Here are typical examples that show how you can obtain an iterator.

```cpp
new MapIterator(myMap)
myMap.getEnumerator()
```

For **Set** objects, if any elements are added or removed after an iterator is created, the iterator instance can no longer be used to read from or step through the collection.

For **Map** objects, as for **Set** objects, if any elements are removed, the iterator is no longer valid. However, a **MapIterator** object remains valid even after a call to the **Map.insert** method, regardless of whether the key is new, or whether the key already exists and only the value is being updated in the **Map** element. Code that calls **Map.insert** and depends on the iterator object remaining valid might fail if it's run as .NET Framework CIL.

You can use the collection classes to form more complex classes. For example, you can easily implement a stack by using a list where elements are always added to the beginning of the list. The newest element then occupies the top of the stack.

You can also extend the collection classes. For example, you can extend the **List** class to create a list of customer records where the operations are type-safe. In this case, the derived collection class will accept only customer records.
This topic describes extended data types in X++.

Extended data types are user-defined types that are based on the boolean, int, int64, real, str, and date primitive data types, and on the container composite type. An EDT is a primitive data type or container that has a supplementary name and additional properties. For example, you can create a new EDT that is named Name and base it on a string. You can then use the new EDT in variable and field declarations in the development environment.

You can also base EDTs on other EDTs. EDTs are standard data types, but they have a specific name and additional properties. EDTs undergo the same value and type conversions as the standard data types that they are based on. Here are the benefits of EDTs:

- Code is easier to read, because variables have a meaningful data type. For example, the data type is Name instead of str.
- The properties that you set for an EDT are used by all instances of that type. Therefore, EDTs help reduce work and promote consistency. For example, account numbers (AccountNum data type) have the same properties throughout the system.
- You can create hierarchies of EDTs. The EDTs can inherit the appropriate properties from the parent, and you can change other properties. For example, the ItemCode data type is used as the basis for the MarkupItemCode and PriceDiscItemCode data types.

Create an EDT

This feature isn't implemented as a language construct. To create an EDT, follow these steps.

1. In Solution Explorer, right-click on the project, point to Add, and then click New item.
2. In the Add New Item dialog box, select Installed and then Artifacts in the left pane.
3. In the middle pane, select the EDT type to create.
4. Enter a name, and then click Add.

EDT example

```java
public void EdtMethod()
{
    // Example of declaring EDT variables where
    // a UserGroupID (integer) variable is declared and initialized to 1.
    UserGroupID groupID = 1;

    // An Amount (real) variable is declared.
    Amount currency;
}
```
This topic describes statements in X++.

Comments

It's a good practice to add comments to your code. Comments make a program easier to read and understand. Comments are ignored when the program is compiled. Your comments can use either the // style or the /* style. However, a best practice is to use the // style for comments, and even for multiline comments.

```//
// This is an example of a comment.
//
/*
/* Here is another example of a comment. */
/*
```

print statements

You use the print statement to output text through System.Diagnostics.WriteLine to the Visual Studio Output window. During testing, the print statement is an alternative to the Global::info method, which shows text in the Infolog window. The following table compares the print statement and the info method.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>PRINT STATEMENT</th>
<th>INFO METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ease of invocation</td>
<td>The print statement automatically converts various data types into strings. It can convert multiple data types in one invocation.</td>
<td>The info method requires that the input parameter be a string.</td>
</tr>
<tr>
<td>Ability to copy contents to the clipboard</td>
<td>Text is easily copied from the Output window to the clipboard.</td>
<td>Text is easily copied from the Infolog window to the clipboard.</td>
</tr>
<tr>
<td>Typical usage</td>
<td>The print statement is used for convenience during testing. It can help you debug small issues without having to run a formal debugger.</td>
<td>The info method is appropriate for use in production.</td>
</tr>
</tbody>
</table>

Example of a print statement

The following code example demonstrates the print statement automatically converting any date type to a string. You do not need to prefix info with Global:: when you call it.
str hello = "Hello";
int fortytwo = 42;
utcDateTime now = DateTimeUtil::utcNow();
Dialog dialog = new Dialog();

print "The print statement automatically converts data types to strings.";
print hello, " -- ", fortytwo, " -- ", now, " -- ", dialog;
// Output to the Print window:
// The print statement automatically converts data types to strings.
// Hello -- 42 -- 10/3/2011 09:18:10 pm -- 1

// int2Str converter is needed when using info().
info("Hello");
info(int2Str(fortytewo));

// Output to Infolog window:
// Hello
// 42

TODO comments
The compiler recognizes the string TODO when it occurs at the start of a comment. The TODO string prompts the compiler to report the rest of the comment text in the Task List window in Microsoft Visual Studio. To open the Task List window, select View, and then select Task Window. The Task Window reports the line number where the TODO comment can be found in the code.

Here are the rules for using TODO in comments:

- The TODO string can appear in a comment that uses either the // style or the /* */ style.
- The TODO string must be the very first non–white space string in the comment. A carriage return, a line feed, a tab, and a space are all considered white space.
- No white space is required between the start of the comment and the TODO.
- The TODO string is case-insensitive. However, the convention is to type TODO in all uppercase letters, instead of ToDo or another variation.
- The TODO string can have any characters appended to it. However, the convention is either to append a colon to the TODO string or to follow it with a white space.
- The rest of the comment after the TODO string is reported as the task description. If the comment is longer than 200 characters, it might appear truncated on the Tasks tab.
- The TODO task description can be spread over multiple lines when the /* */ comment style is used.

Examples of TODO comments
The following examples show TODO comments.

// An example of using TODO in the // style of comment.
public boolean isLate()
{
    // TODO: Finish this stub.
    return true;
}

// An example of using TODO in the /* */ style of comment.
public boolean isLate()
{
    /* TODO Finish this stub */
    return true;
}
Unsupported statements: changeSite, pause, and window
The changeSite , pause , and window keywords are no longer a part of the X++ language. These keywords will
cause compilation errors if you use them.

Ignored statements: server and client
In previous versions (AX2012 and earlier), you could designate a method to run on either the client or the server.
This is no longer possible, because all X++ code is executed as .NET CIL on the server. The keywords client and
server are ignored. Their use doesn't cause a compile error, but they should not be used in any new X++ code.

using clauses
You use using clauses so that you don't have to provide the fully qualified name of a type. The using clause
must precede the class that it applies, and it's required in every source file that you want it to apply to. Typically,
all using clauses are put at the beginning of the source file. You can also provide aliases that introduce a short
name for a fully qualified name. Aliases can denote namespaces or classes.
The following example shows a using clause, a namespace alias, and a class alias.
using System;
using IONS=System.IO; // Namespace alias
using Alist=System.Collections.ArrayList; // Class alias
class UsingClass
{
public static void test()
{
Int32 I;
// Alternative to System.Int32
Alist al = new Alist(); // Using a class alias
al.Add(1);
str s = IONS.Path::ChangeExtension(@"c:\tmp\test.xml", ".txt"); // Using a namespace alias
}
}

using statements
The using statement helps guarantee that objects that implement IDisposable are disposed of correctly. When
you use an IDisposable object, you should declare and instantiate it in a using statement. The using statement
calls the Dispose method on the object in the correct way, even if an exception occurs while you're calling
methods on the object. You can achieve the same result by putting the object inside a tr y block and then
explicitly calling Dispose in a finally block. The using statement simplifies the syntax and disposes of the
object correctly. Here is the syntax for a using statement:
using ( expression ) { statement }
In this syntax, statement can be a block of statements, and expression declares and instantiates an object that
implements IDisposable . The following example creates and uses a StreamReader object.
static void AnotherMethod()
{
str textFromFile;
using (System.IO.StreamReader sr = new System.IO.StreamReader("c:\\test.txt"))
{
textFromFile = sr.ReadToEnd();
}
}


This topic describes conditional statements in X++. The conditional statements are if, if...else, switch, and the ternary operator (?). You use conditional statements to specify whether a block of code is executed. Different conditional statements offer advantages in different situations.

### if and if...else statements

The if statement evaluates a conditional expression, and then executes a statement or a set of statements if the conditional expression is evaluated as true. You can use the else clause to provide an alternative statement or set of statements that is executed if the condition is evaluated as false. The syntax for an if...else statement is:

```plaintext
if (expression) statement [else statement]
```

In this syntax, both occurrences of statement can be compound statements (statements enclosed in braces). The expression in the parentheses (the conditional expression) can be any valid expression that is evaluated as true or false. All numbers except 0 (zero) are true. All non-empty strings are true. You can nest if statements. However, if the nesting of if statements becomes too deep, you should consider using a switch statement instead.

**Examples of if and if...else statements**

```plaintext
// if statement
if (a > 4)
{
    info("a is greater than 4");
}

// if... else statement
if (a > 4)
{
    info("a is greater than 4");
} else
{
    info("a is less than or equal to 4");
}
```

### switch statements

The switch statement is a multibranch language construct that has the same behavior as nested if. The expression of the switch statement is evaluated and checked against each case value. The case values must be constants that the compiler can evaluate.

- If a case constant matches the switch expression, the case statement is executed.
- If the case contains a break statement, the program then jumps out of the switch.
- If the case doesn't contain a break statement, the program continues and executes the next case statements.
- If no matches are found, the default statement is executed.
- If there are no matches and no default statement, none of the statements inside the switch statement are executed.

Here is the syntax for a switch statement:
The syntax for a `case` statement is:

```plaintext
case expression { , expression } : statement
```

In the syntax for both a `switch` statement and a `case` statement, every occurrence of `statement` can be replaced with a block of statements by enclosing the block in braces ({}).

**Examples of switch statements**

When you include the `break` keyword in a switch statement, the execution of the case branch terminates, and the statement following the switch is executed. As shown in the following example, if the Debtor account number is 1000, the program executes "do something", and then continues execution after the switch statement.

```plaintext
switch (Debtor.AccountNo)
{
    case "1000":
        // do something
        break;
    case "2000":
        // do something else
        break;
    default:
        // default statement
        break;
}
```

The following code examples makes the execution drop through the first case branch by omitting a `break` statement. If x is 10, b is assigned to a, and d is assigned to c. If x is 11, d is assigned to c. If x is 12, f is assigned to e.

```plaintext
switch (x)
{
    case 10:
        a = b;
    case 11:
        c = d;
        break;
    case 12:
        e = f;
        break;
}
```

If you do not use the `break` statement, the program flow in the switch statement continues into the next case. Code segments A and B have the same behavior.

```plaintext
// Code segment A (break omitted)
case 13:
    case 17:
    case 21:
        case 500:
            info("g");
            break;

// Code segment B (the values are comma-delimited)
case 13, 17, 21, 500;
    info("g");
    break;
```
Ternary operator (?)

The ternary operator (?) is a conditional statement that is resolved to one of two expressions. The result can be assigned to a variable. By contrast, an if statement provides conditional branching of the program flow but can’t be assigned to a variable. Here is the syntax for the ternary operator:

\[ \text{expression1} ? \text{expression2} : \text{expression3} \]

In this syntax, expression1 must return a value of true or false. If expression1 is true, the whole ternary statement returns expression2. Otherwise, the statement returns expression3. Both expression2 and expression3 must have the same type.

Examples of the ternary operator (?)

The following code example returns one of two strings based on a Boolean return value from a method call. The Boolean expression indicates whether the CustTable table has a row with a RecId field value of 1. If this Boolean expression is true (meaning RecId != 0), found is assigned to result. Otherwise, the alternative not found is assigned to result.

```
result = (custTable::find("1").RecId) ? "found" : "not found";
```

You can nest statements with the ternary operator. The following example assigns one of three values to level based on the value of x.

```
int x = 1001;
str level = x <= 1000 ? "A" : (x <= 2000 ? "B" : "C");
info(level);
// Output is "B".
```
This topic describes loop statements in X++.  

There are three loop statements: **for**, **while**, and **do...while**. A loop repeats its statement until the condition that is set for the loop is **false**. Within the loop statements, you can use **break** and **continue** statements.

### for loops

The syntax of a **for** loop is:

```csharp
for ( initialization ; test ; increment ) { statement }
```

The **for** loop repeatedly executes the **statement** for as long as the conditional expression **test** is **true**. **statement** can be a block of statements. The body of the **for** loop (**statement**) might be executed zero or more times, depending on the results of **test**.

A **for** loop differs from other loops because an initial value can be assigned to a control variable, and because there is a statement for incrementing or decrementing the variable. These additions make a **for** loop especially useful for traversing lists, containers, and arrays because they have a fixed number of elements.

You can also apply a statement to each element and increment your way through the elements, setting the condition to test for the last element.

**Example of a for loop**

In the following code example, the items in an array of integers are printed.

```csharp
int integers[10];
for (int i = 0; i < 10; i++)
{
    info(int2str(integers[i]));
}
// The output is a series of 0's.
```

### while loops

The syntax of a **while** loop is:

```csharp
while ( expression ) statement
```

A **while** loop repeatedly executes **statement** for as long as the conditional **expression** is **true**. **statement** can be replaced by a block of statements. **statement** is executed as many times as the condition is met (zero to many).

**Example of a while loop**

The following code example demonstrates a **while** loop that traverses a container and prints out the contents of the container.
container cont = ["one", "two", "three"];  
int no = 1;  
while (no <= conlen(cont))  
{  
    info(conPeek(cont, no));  
    no++;  
}  
// The output is "one", "two", "three".

do...while loops
The syntax of the do...while loop is:

```
do { statement } while ( expression );
```

The do...while loop is similar to the while loop, but the condition appears after the statement that must be executed. The statement can be a block of statements. The statement is always executed at least one time, because the condition is tested after the statement is executed. The do...while loop is well-suited to tasks that must always be done at least one time, such as getting parameters for a report.

Example of a do...while loop
The following code example finds the smallest power of 10 that is larger than `realNumber`.

```
int FindPower(real realNumber)  
{  
    int exponent = -1;  
    real curVal;  
    do  
    {  
        exponent++;  
        curVal = power(10, exponent);  
    }  
    while (realNumber > curVal);  
    return exponent;  
}
```

continue statement
The continue statement causes execution to move directly to the next iteration of a for, while, or do...while loop. For do or while, the test is executed immediately. For a for statement, the increment step is executed.

Example of a continue statement
In the following code example, if `Iarray[i] <= a`, the remaining statements in the loop are not executed, and `i` is incremented before the if statement is tried again.
int Iarray[100];
for (int i = 0; i < 100; i++)
{
    if (Iarray[i] <= 0)
    {
        Info("Will continue.");
        continue;
    }
    info("Did not continue.");
}
// The output is "Will continue." for all 100 iterations.

break statement

The **break** statement within a loop is used to terminate that loop. Execution then moves to the first statement after the loop.

**Example of a break statement**

This example is uses a **break** statement within a **while** loop. When used within a loop, the loop is terminated and execution continues from the statement following the loop. This works for **do...while** and **for** loops as well.

```java
var mainMenu = SysDictMenu::newMainMenu();
var enum = mainMenu.getEnumerator();
var found = false;
while (enum.moveNext())
{
    var menuItem = enum.current();
    if (menuItem.label() == "StringOfInterest")
    {
        found = true;
        break;
    }
}
if (found)
{
    // do something
}
```
This topic describes exception handling in X++. You handle errors by using the `throw`, `try...catch`, `finally`, and `retry` statements to generate and handle exceptions.

An exception is a regulated jump away from the sequence of program execution. The instruction where program execution resumes is determined by `try...catch` blocks and the type of exception that is thrown. An exception is represented by a value of the `Exception` enumeration, or an instance of .NET's `System.Exception` class or a derived class. One exception that is often thrown is the `Exception::error` enum value. A common practice is to write diagnostic information to the Infolog before the exception is thrown.

The `Global::error` method is often the best way to write diagnostic information to the Infolog. For example, your method might receive an input parameter value that isn't valid. In this case, the method can throw an exception to immediately transfer control to a `catch` code block that contains logic for handling this error situation. You don't necessarily have to know the location of the `catch` block that will receive control when the exception is thrown.

### throw statements

You use the `throw` keyword to throw an `Exception` enum value. For example, the following statement throws an error exception.

```plaintext
throw Exception::error;
```

Instead of throwing an enum value, a best practice is to use the output of the `Global::error` method as the operand for `throw`.

```plaintext
throw Global::error("The parameter value is invalid.");
```

The `Global::error` method can automatically convert a label into the corresponding text. This functionality helps you write code that can be localized more easily.

```plaintext
throw Global::error("@SYS98765");
```

The static methods on the `Global` class can be called without the `Global::` prefix. For example, the `Global::error` method can be called like this.

```plaintext
error("My message.");
```

In Platform update 31 or later versions, the `throw` keyword can be used to throw .NET exceptions.

```plaintext
throw new InvalidOperationException("This function is not allowed");
```

Also in Platform update 31 or later, the `throw` keyword can be used by itself inside a `catch` block. In such a case, `throw` will behave like the `rethrow` statement in C#. The original exception, exception message and its context such as call stack will be rethrown and be available to any catch statements in calling code.
try
{
    throw Exception::error;
}
catch
{
    // locally handle exception
    // then rethrow for caller
    throw;
}

try, catch, finally, and retry statements

When an exception is thrown, it's first processed through the catch list of the innermost try block. If a catch block is found that handles the kind of exception that is being thrown, program control jumps to that catch block. If the catch list has no block that specifies the exception, the system passes the exception to the catch list of the next-innermost try block. The catch statements are processed in the same sequence as they appear in the code.

It's a common practice to have the first catch statement handle the Exception::Error enum value. One strategy is to have the last catch statement leave the exception type unspecified. In this case, the last catch statement handles all exceptions that aren't handled by any earlier catch statement. This strategy is appropriate for the outermost try..catch blocks.

An optional finally clause can be included in try..catch statements. The semantics of a finally clause are the same as they are in C#. The statements in the finally clause are executed when control leaves the try block, either normally or through an exception.

The retry statement can be written only in a catch block. The retry statement causes control to jump up to the first line of code in the associated try block. The retry statement is used when the cause of the exception can be fixed by the code in the catch block. The retry statement gives the code in the try block another opportunity to succeed. The retry statement erases all messages that have been written to the Infolog since program control entered the try block.

NOTE
You must make sure that your retry statements don't cause an infinite loop. As a best practice, the try block should include a variable that you can test to find out whether you're in a loop.

try
{
    // Code here.
}
catch (Exception::Numeric)
{
    info("Caught a Numeric exception.");
}
catch
{
    info("Caught an exception.");
}
finally
{
    // Executed no matter how the try block exits.
}

The system exception handler
If no `catch` statement handles the exception, it's handled by the system exception handler. The system exception handler doesn't write to the Infolog. Therefore, an unhandled exception can be hard to diagnose. We recommended that you follow all these guidelines to provide effective exception handling:

- Have a `try` block that contains all your statements in the outermost frame on the call stack.
- Have an unqualified `catch` block at the end of your outermost `catch` list.
- Avoid throwing an `Exception` enum value directly.
- Throw the enum value that is returned from one of the following methods on the `Global` class:
  - `Global::error`, `Global::warning`, or `Global::info`. (You can omit the implicit `Global::` prefix).
- When you catch an exception that hasn't been shown in the Infolog, call the `Global::info` function to show it.

`Exception::CLRError`, `Exception::UpdateConflictNotRecovered`, and system kernel exceptions are examples of exceptions that aren't automatically shown in the Infolog.

**Exceptions and CLR interop**

You can call Microsoft .NET Framework classes and methods that reside in assemblies that are managed by the common language runtime (CLR). When a .NET Framework `System.Exception` instance is thrown, your code can catch it by declaring a variable of type `System.Exception` to catch any .NET exception, or one of its derived classes to catch a specific .NET exception type as shown in the following example.

```csharp
System.ArgumentException ex;
try
{
    throw new System.ArgumentException("Invalid argument specified");
}
catch(ex)
{
    error(ex.Message);
}
```

In releases prior to Platform update 31, .NET exceptions can be caught by referencing `Exception::CLRError`. Your code can obtain a reference to the `System.Exception` instance by calling the `CLRInterop::getLastException` method.

```csharp
try
{
    // call to .NET code which throws exception
}
catch(Exception::CLRError)
{
    System.Exception ex = CLRInterop::getLastException();
    error(ex.Message);
}
```

**Ensuring that exceptions are shown**

Exceptions of the `Exception::CLRError` type aren't shown in the Infolog, because these exceptions aren't issued by a call to a method such as `Global::error`. In your `catch` block, your code can call `Global::error` to report the specific exception.

**Global class methods**

This section describes some `Global` class methods in more detail. These class methods include `Global::error`, `Global::info`, and `Global::exceptionTextFallThrough`.

**Global::error method**

The following code shows how the `error` method is declared.
static Exception error
(SysInfoLogStr txt,
URL helpURL = '',
SysInfoAction _sysInfoAction = null)

The return type is the Exception::Error enum value. The error method doesn't throw an exception. It just
provides an enum value that can be used in a throw statement. The throw statement throws the exception.
Here are descriptions of the parameters for the error method. Only the first parameter is required.

- SysInfoLogStr txt is a str of the message text. It can also be a label reference, such as
  strFmt("@SYS12345", strThingName).
- The URL helpUrl is a reference to the location of a Help topic in Application Explorer, such as
  "KernDoc:\Functions\\substr". The parameter value is ignored if _sysInfoAction is supplied.
- The SysInfoAction is an instance of a class that extends the SysInfoAction class. The method overrides
  that we recommend for the child class are the description method, the run method, the pack method, and
  the unpack method.

Global::info method
The Global::info method is often used to show text in the Infolog. In programs, it's often written as info("My
message."). Although the info method returns an Exception::Info enum value, you will rarely want to throw
Exception::Info, because nothing unexpected has occurred.

Global::exceptionTextFallThrough method
Occasionally, you want to do nothing inside your catch block. However, the X++ compiler generates a warning
if you have an empty catch block. To avoid this warning, call the Global::exceptionTextFallThrough method
in the catch block. The method does nothing, but it satisfies the compiler and explicitly states the intention.

Exceptions inside transactions
If an exception is thrown inside a transaction, the transaction is automatically canceled (that is, a ttsAbort
operation occurs). This behavior applies for both exceptions that are thrown manually and exceptions that the
system throws. When an exception is thrown inside a ttsBegin-ttsCommit transaction block, no catch
statement inside that transaction block can process the exception, (unless it is a UpdateConflict or a
DuplicateKeyException). Instead, the innermost catch statements that are outside the transaction block are
the first catch statements that are tested.

The finally clause will be executed even in transaction scope.

Exceptions and using statements
The semantics of using statements are not impacted by exception scope.

using (var athing = new SomethingDisposable())
{
    // Do work.
}

Is exactly the same as:
Examples of exception handling

Showing exceptions in the Infolog

The following code example shows exceptions in the Infolog.

```c++
// This example shows that a direct throw of Exception::Error does not
// display a message in the Infolog. This is why we recommend the
// Global::error method.
static void TryCatchThrowError1Job(Args _args)
{
    /***
    * The 'throw' does not directly add a message to the Infolog.
    * The exception is caught.
    ***/
    try
    {
        info("In the 'try' block. (j1)");
        throw Exception::Error;
    }
    catch (Exception::Error)
    {
        info("Caught 'Exception::Error'.");
    }

    /**********  Actual Infolog output
    Message (03:43:45 pm)
    In the 'try' block. (j1)
    Caught 'Exception::Error'.
    **********/
}
```

Using the error method to write exception information to the Infolog

The following code example uses the `error` method to write exception information to the Infolog.

```c++
var athing = new SomethingDisposable();
try
{
    // Do work.
}
finally
{
    if (athing != null)
        athing.Dispose();
}```
// This example shows that the use of the Global::error method
// is a reliable way to display exceptions in the Infolog.
static void TryCatchGlobalError2Job(Args _args)
{
    /***
    The 'Global::error()' does directly add a message to the Infolog.
    The exception is caught.
    ***/
    try
    {
        info("In the 'try' block. (j2)" subtract 1); // Adjusted
        throw Global::error("Written to the Infolog.");
    }
    catch (Exception::Error)
    {
        info("Caught 'Exception::Error'.");
    }

    /***
    Infolog output
    Message (03:51:44 pm)
    In the 'try' block. (j2)
    Written to the Infolog.
    Caught 'Exception::Error'.
    ***/
}

**Handling a CLRError**

The following code example handles a CLRError exception.
// This example shows that a CLRError exception is not displayed
// in the Infolog unless you catch the exception and manually
// call the info method. The use of the CLRInterop::getLastException
// method is also demonstrated.
static void TryCatchCauseCLRError3Job(Args _args)
{
    /***
    The 'netString.Substring(-2)' causes a CLRError,
    but it does not directly add a message to the Infolog.
    The exception is caught.
    ***/
    System.String netString = "Net string."
    System.Exception netExcepn;
    try
    {
        info("In the 'try' block. (j3)" );
        netString.Substring(-2); // Causes CLR Exception.
    } catch (Exception::Error)
    {
        info("Caught 'Exception::Error'.");
    } catch (Exception::CLRError)
    {
        info("Caught 'Exception::CLRError'.");
        netExcepn = CLRInterop::getLastException();
        info(netExcepn.ToString());
    }

    /**********  Actual Infolog output (truncated for display)
    Message (03:55:10 pm)
    In the 'try' block. (j3)
    Caught 'Exception::CLRError'.
    System.Reflection.TargetInvocationException: Exception has been thrown by the target of an invocation. --->
    System.ArgumentOutOfRangeException: StartIndex cannot be less than zero.
    Parameter name: startIndex
    at System.String.InternalSubStringWithChecks(Int32 startIndex, Int32 length, Boolean fAlwaysCopy)
    at System.String.Substring(Int32 startIndex)
    at ClrBridgeImpl.InvokeClrInstanceMethod(ClrBridgeImpl* , ObjectWrapper* objectWrapper, Char* pszMethodName,
    Int32 argsLength, ObjectWrapper** arguments, Boolean* argsAreByRef, Boolean* isException)
    **********/}

Using a retry statement
The following code example uses a retry statement.
// This example shows how to use the retry statement. The print statements are included because retry causes earlier Infolog messages to be erased.

static void TryCatchRetry4Job(Args _args)
{
    /***
    Demonstration of 'retry'. The Infolog output is partially erased by 'retry', but the Print window is fully displayed.
    ***/
    Exception excepnEnum;
    int nCounter = 0;
    try
    {
        info(" .");
        print(" .");
        info("In the 'try' block, [" + int2str(nCounter) + "]. (j4)"");
        print("In the 'try' block, [" + int2str(nCounter) + "]. (j4)"");
        nCounter++;
        if (nCounter >= 3) // Prevent infinite loop.
        {
            info("---- Will now throw a warning, which is not caught.");
            print("---- Will now throw a warning, which is not caught.");
            throw Global::warning("This warning will not be caught. [" + int2str(nCounter) + "]");
        }
        else
        {
            info("Did not throw a warning this loop. [" + int2str(nCounter) + "]");
            print("Did not throw a warning this loop. [" + int2str(nCounter) + "]");
        }
        excepnEnum = Global::error("This error message is written to the Infolog.");
        throw excepnEnum;
    }
    catch (Exception::Error)
    {
        info("Caught 'Exception::Error'.");
        print("Caught 'Exception::Error'.");
        retry;
    }
    info("End of job.");
    print("End of job.");

    /***********  Actual Infolog output*/
    Message (04:33:56 pm)
    .
    In the 'try' block, [2]. (j4)
    ---- Will now throw a warning, which is not caught.
    This warning will not be caught. [3]
    **********/}

**Throwing an exception inside a transaction**

The following code example throws an exception in a transaction block.
This example uses three levels of try nesting to illustrate where an exception is caught when the exception is thrown inside a ttsBegin-ttsCommit transaction block.

```csharp
static void TryCatchTransaction5Job(Args _args) {
    /***
    Shows an exception that is thrown inside a ttsBegin - ttsCommit
    transaction block cannot be caught inside that block.
    ***/
    try
    {
        try
        {
            ttsbegin;
            try
            {
                throw error("Throwing exception inside transaction.");
            }
            catch (Exception::Error)
            {
                info("Catch_1: Unexpected, caught in 'catch' inside the transaction block.");
            }
            ttscommit;
        }
        catch (Exception::Error)
        {
            info("Catch_2: Expected, caught in the innermost 'catch' that is outside of the transaction block.");
        }
    }
    catch (Exception::Error)
    {
        info("Catch_3: Unexpected, caught in 'catch' far outside the transaction block.");
    }
    info("End of job.");
}
```

Using Global::error with a SysInfoAction parameter

When your code throws an exception, it can write messages to the Infolog. You can make those Infolog messages more helpful by using the SysInfoAction class.

In the following example, a SysInfoAction parameter is passed in to the Global::error method. The error method writes the message to the Infolog. When the user double-clicks the Infolog message, the SysInfoAction.run method is run.

In the run method, you can write code that helps diagnose or fix the issue that caused the exception. The object that is passed in to the Global::error method is constructed from a class that you write that extends SysInfoAction.

The following code sample is shown in two parts.

- The first part shows a job that calls the Global::error method and then throws the returned value. An instance of the SysInfoAction_PrintWindow_Demo class is passed in to the error method.
- The second part shows the SysInfoAction_PrintWindow_Demo class.

Part 1: Calling Global::error
static void Job_SysInfoAction(Args _args)
{
    try
    {
        throw Global::error
        ("Click me to make the Print window display."
         ,"
         ,new SysInfoAction_PrintWindow_Demo()
         );
    } catch
    {
        warning("Issuing a warning from the catch block.");
    }
}

Part 2: The SysInfoAction_PrintWindow_Demo class

public class SysInfoAction_PrintWindow_Demo extends SysInfoAction
{
    str m_sGreeting; // In classDeclaration.
    public str description()
    {
        return "Starts the Print Window for demonstration.";
    }
    public void run()
    {
        print("This appears in the Print window.");
        print(m_sGreeting);
        /*********** Actual Infolog output
        Message (03:19:28 pm)
        Click me to make the Print window display.
        Issuing a warning from the catch block.
        *************/
    }
    public container pack()
    {
        return ["Packed greeting."]; // Literal container.
    }
    public boolean unpack(container packedClass, Object object = null)
    {
        [m_sGreeting] = packedClass;
        return true;
    }
}

List of exceptions

The following table shows the exception literals that are the values of the Exception enumeration.

<table>
<thead>
<tr>
<th>EXCEPTION LITERAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Break</td>
<td>The user pressed Break or Ctrl+C.</td>
</tr>
<tr>
<td>CLRError</td>
<td>An error occurred while the CLR functionality was being used.</td>
</tr>
<tr>
<td>CodeAccessSecurity</td>
<td>An error occurred while the CodeAccessPermission.demand method was being used.</td>
</tr>
<tr>
<td>EXCEPTION LITERAL</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>DDEerror</td>
<td>An error occurred while the <strong>DDE</strong> system class was being used.</td>
</tr>
<tr>
<td>Deadlock</td>
<td>A database deadlock occurred, because several transactions are waiting for each other.</td>
</tr>
<tr>
<td>DuplicateKeyException</td>
<td>An error occurred in a transaction that is using Optimistic Concurrency Control. The transaction can be retried (use a <code>retry</code> statement in the <code>catch</code> block).</td>
</tr>
<tr>
<td>DuplicateKeyExceptionNotRecovered</td>
<td>An error occurred in a transaction that is using Optimistic Concurrency Control. The code won't be retried. This exception can't be caught inside a transaction.</td>
</tr>
<tr>
<td>Error</td>
<td>A fatal error occurred. The transaction has been stopped.</td>
</tr>
<tr>
<td>Info</td>
<td>This exception literal holds a message for the user. Don't throw an <strong>info</strong> exception.</td>
</tr>
<tr>
<td>Internal</td>
<td>An internal error occurred in the development system.</td>
</tr>
<tr>
<td>Numeric</td>
<td>An error occurred while the <code>str2int</code>, <code>str2int64</code>, or <code>str2num</code> function was being used.</td>
</tr>
<tr>
<td>Sequence</td>
<td></td>
</tr>
<tr>
<td>UpdateConflict</td>
<td>An error occurred in a transaction that is using Optimistic Concurrency Control. The transaction can be retried (use a <code>retry</code> statement in the <code>catch</code> block).</td>
</tr>
<tr>
<td>UpdateConflictNotRecovered</td>
<td>An error occurred in a transaction that is using Optimistic Concurrency Control. The code won't be retried. This exception can't be caught within a transaction.</td>
</tr>
<tr>
<td>Warning</td>
<td>An exceptional event has occurred. Although the user might have to take action, the event isn't fatal. Don't throw a <strong>warning</strong> exception.</td>
</tr>
<tr>
<td>SQL connection error X++ exception</td>
<td>An error occurred when during the query execution. The transaction will be canceled. This exception can't be caught within a transaction.</td>
</tr>
</tbody>
</table>
This topic describes the SQL connection error exception types in X++.

**TransientSqlConnectionError X++ exception**

During an X++ SQL query execution, when a transient SQL connection error occurs on the server side, a TransientSqlConnectionError X++ exception will occur. Depending on the application requirements, the application should catch and handle the exception.

This exception usually occurs during a large transaction or when the database is under a lot of processing pressure.

The TransientSqlConnectionError exception is not catchable within the transaction. The X++ transaction that encounters this exception is canceled (calling `ttsAbort`) before the exception occurs. This means that you need to use the catch block to identify the transient SQL connection error instead of a generic X++ error exception, and then retry the outermost transaction or retry application code logic in a new session. This exception allows the application to be designed for transient server failures.

If an application transaction takes a long time to process, you can use multiple incremental delays to catch the TransientSqlConnectionError exception. Retrying your application code in a new session is most likely to succeed after you have caught the exception.

**Example**

```java
public static void LargeTransactionWrapper()
{
    try
    {
        LargeTransaction();
    }
    catch (Exception::TransientSqlConnectionError)
    {
        info("Caught transient SQL connection error, ttslevel=" + int2Str(appl.ttsLevel()));
        // At this point, transaction is canceled
        // Code that indicates retry is possible
    }
    finally
    {
        // Do clean up
    }
}
```
This topic describes the operators supported in X++.

**Assignment operators**

An assignment changes the value of a variable or field. The following table shows the X++ assignment operators. There is no difference between prefix and postfix operators.

<table>
<thead>
<tr>
<th>OPERATOR</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Assign the expression on the right of the equal sign to the variable on the left.</td>
</tr>
<tr>
<td>+=</td>
<td>Assign the current variable value plus the expression on the right to the variable on the left.</td>
</tr>
<tr>
<td>++</td>
<td>Increment the variable by 1.</td>
</tr>
<tr>
<td>-=</td>
<td>Assign the current variable value minus the expression on the right to the variable on the left.</td>
</tr>
<tr>
<td>--</td>
<td>Decrement the variable by 1.</td>
</tr>
</tbody>
</table>

**Code examples for assignment operators**
// An example of assignment operators and their output.
static void Example1()
{
    int i = 1;
    // Using the = operator. i is assigned the value of i, plus 1. i = 2.
    i = i + 1;
    info(strFmt("Example 1: The result is "), i); // The result is 2.
}

static void Example2()
{
    int i = 1;
    // Using the += operator. i is assigned the value of i, plus 1.
    // i = 2 (i = i + 1).
    i += 1;
    info(strFmt("Example 2: The result is "), i); // The result is 2.
}

static void Example3()
{
    int i = 1;
    // Using the ++ operator. i is incremented by 1, and then
    // by 1 again in the second statement. The final value of i is 3.
    i++; ++i;
    info(strFmt("Example 3: The result is "), i); // The result is 3.
}

static void Example4()
{
    int i = 1;
    // Using the -= operator. i is assigned the value of i minus 1.
    // i = 0 (i = i - 1).
    i -= 1;
    info(strFmt("Example 4: The result is "), i); // The result is 0.
}

static void Example5()
{
    int i = 1;
    // Using the -- operator. i is decremented by 1, and then by
    // 1 again in the second statement. The final value of i is -1.
    i--; --i;
    info(strFmt("Example 5: The result is "), i); // The result is -1.
}

Arithmetic operators

You use arithmetic operators to perform numeric calculations. Most of the operators are binary and take two operands. However, the not (~) operator is unary and takes only one operand. Syntax for binary operators: `expression1 ArithmeticOperator expression2` Syntax for unary operators: `ArithmeticOperator expression1`

<table>
<thead>
<tr>
<th>OPERATOR</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&lt;</td>
<td>The left shift operator performs <code>expression2</code> left shift (multiplication by 2) on <code>expression1</code>.</td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>The right shift operator performs <code>expression2</code> right shift (division by 2) on <code>expression1</code>.</td>
</tr>
<tr>
<td>OPERATOR</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>*</code></td>
<td>The <strong>multiply</strong> operator multiplies <code>expression1</code> by <code>expression2</code>.</td>
</tr>
<tr>
<td><code>/</code></td>
<td>The <strong>divide</strong> operator divides <code>expression1</code> by <code>expression2</code>.</td>
</tr>
<tr>
<td><code>DIV</code></td>
<td>The <strong>integer division</strong> operator performs an integer division of <code>expression1</code> by <code>expression2</code>.</td>
</tr>
<tr>
<td><code>MOD</code></td>
<td>The <strong>integer remainder</strong> operator returns the remainder of an integer division of <code>expression1</code> by <code>expression2</code>.</td>
</tr>
<tr>
<td><code>~</code></td>
<td>The <strong>not</strong> operator, or unary operator, performs a binary not operation.</td>
</tr>
<tr>
<td><code>&amp;</code></td>
<td>The <strong>binary AND</strong> operator performs a binary and operation on <code>expression1</code> and <code>expression2</code>.</td>
</tr>
<tr>
<td><code>^</code></td>
<td>The <strong>binary XOR</strong> operator performs a binary XOR-operation on <code>expression1</code> and <code>expression2</code>.</td>
</tr>
<tr>
<td>`</td>
<td>`</td>
</tr>
<tr>
<td><code>+</code></td>
<td>The <strong>plus</strong> operator adds <code>expression1</code> to <code>expression2</code>.</td>
</tr>
<tr>
<td><code>-</code></td>
<td>The <strong>minus</strong> operator subtracts <code>expression2</code> from <code>expression1</code>.</td>
</tr>
<tr>
<td><code>?</code></td>
<td>The <strong>ternary</strong> operator takes three expressions: <code>expression1</code>? <code>expression2</code>: <code>expression3</code>. If <code>expression1</code> is true, <code>expression2</code> is returned. Otherwise, <code>expression3</code> is returned.</td>
</tr>
</tbody>
</table>

### Code examples for arithmetic operators

```c
int a = 1 << 4; // Perform four left shifts on 1 (1*2*2*2*2). a=16.
int b = 16 >> 4; // Perform four right shifts on 16 (16/2/2/2/2). b=1.
int c = 4 * 5; // Multiply 4 by 5. c=20.
int d = 20 / 5; // Divide 20 by 5. d=4.
int e = 100 div 21; // Return the integer division of 100 by 21. e=4 (4*21 = 84, remainder 16).
int f = 100 mod 21; // Return the remainder of the integer division of 100 by 21. f=16.
int g = ~1; // Binary negate 1 (all bits are reversed). g=-2.
int h = 1 & 3; // Binary AND. Return the bits that are in common in the two integers. h=1.
int i = 1 | 3; // Binary OR. Return the bits that are set in either 1 or 3. i=3.
int j = 1 ^ 3; // Binary XOR. Return the bits that are set in 1 and NOT set in 3, and vice versa. j=2.
int k = 1 + 3; // Add 1 and 3. k=4.
int l = 3 - 1; // Subtract 1 from 3. l=2.
int m = (400 > 4) ? 1 : 5; // If 400>4, 1 is returned. Otherwise, 5 is returned. Because 400>4, 1 is returned. m=1.
```

### Expression operators

The `as` and `is` expression operators control downcast assignments. Downcast assignments involve class or table inheritance. Assignment statements that implicitly downcast can cause errors that are difficult to predict and diagnose. You can use the `as` keyword to make your downcasts explicit. You can use the `is` keyword to
test whether a downcast is valid at run time.

**The as keyword**

Use the `as` keyword for assignments that downcast from a base class variable to a derived class variable. The `as` keyword tells other programmers and the compiler that you believe that the downcast will be valid during run time.

- The compiler reports an error for downcast assignment statements that lack the `as` keyword.
- At run time, the `as` keyword causes the downcast assignment statement to assign `null` if the downcast isn't valid.
- This `is` keyword is often used to safely test whether the `as` keyword will work.

**Code example for the as keyword**

In the following code example, the `DerivedClass` class extends the `BaseClass` class. The code example contains two valid assignments between its `basec` and `derivedc` variables. The upcast assignment to `basec` doesn't require the `as` keyword, but the downcast assignment to `derivedc` does require the `as` keyword. The following code will compile and run without errors.

```csharp
static void AsKeywordExample()
{
    // DerivedClass extends BaseClass.
    BaseClass basec;
    DerivedClass derivedc;
    // BottomClass extends DerivedClass.
    BottomClass bottomc;
    derivedc = new DerivedClass();
    // AS is not required for an upcast assignment like this.
    basec = derivedc;
    // AS is required for a downcast assignment like this.
    derivedc = basec as DerivedClass;
    bottomc = new BottomClass();
    // AS causes this invalid downcast to assign null.
    bottomc = basec as DerivedClass;
}
```

**The is keyword**

The `is` keyword verifies whether an object is a subtype of a specified class. The `is` expression returns `true` if the object is a subtype of the class, or if the object is the same type as the class. The compiler reports an error if an `is` keyword expression compares two types, but neither type is a subtype of the other, and they aren't of the same type. The compiler reports a similar error for any plain assignment statement between two types, where neither type is a subtype of the other, and they aren't of the same type. At run time, the type of variable that references the underlying object is irrelevant to the `is` keyword. The `is` keyword causes the system to verify the object that the variable references, not the declared type of the variable that references the object.

**Code examples for the is keyword**

The following code examples illustrate the conditions that control whether an `is` expression returns `true` or `false`. The code examples depend on the fact that the `Form` class and the `Query` class both extend the `TreeNode` class.
// The compiler issues an error for the following code.
// The compiler ascertains that the Form class and the Query class are not
// part of the same inheritance hierarchy. Both the Form class and the Query class
// extend the TreeNode class, but neither Form nor Query is a subtype of the other.
Form myForm = new Form();
info(strFmt("%1", (myForm is Query)));

// The Infolog displays 0 during run time, where 0 means false. No supertype
// object can be considered to also be of its subtype class.
TreeNode myTreeNode = new TreeNode();
info(strFmt("%1", (myTreeNode is Form)));

// The Infolog displays 0 during run time, where 0 means false. A null
// reference causes the is expression to return false.
Form myForm;
info(strFmt("%1", (myForm is Form)));

// The Infolog displays 1 during run time, where 1 means true.
// An object is an instance of its own class type.
Form myForm = new Form();
info(strFmt("%1", (myForm is Form)));

// The Infolog displays 1 during run time, where 1 means true.
// Every subtype is also of its supertype.
Form myForm = new Form();
info(strFmt("%1", (myForm is TreeNode)));

// The Infolog displays 1 during run time, where 1 means true.
// The type of the underlying object matters in the is expression,
// not the type of the variable that references the object.
Form myForm = new Form();
TreeNode myTreeNode;
myTreeNode = myForm; // Upcast.
info(strFmt("%1", (myTreeNode is Form)));

Code example for the is and as keywords
The following code example contains a typical use of the is keyword. The as keyword is used after the is keyword verifies that the as keyword will succeed. In this example, the is and as keywords are uppercase to make them more visible.

```csharp
static void IsKeywordExample()
{
    DerivedClass derivedc;
    BaseClass basec;
    basec = new DerivedClass(); // An upcast.
    if (basec IS DerivedClass)
    {
        info("Test 1: (basec IS DerivedClass) is true. Good.");
        derivedc = basec AS DerivedClass;
    }
    basec = new BaseClass();
    if (!(basec IS DerivedClass))
    {
        info("Test 2: !(basec IS DerivedClass) is true. Good.");
    }
}

// Output to the Infolog
Test 1: (basec IS DerivedClass) is true. Good.
Test 2: !(basec IS DerivedClass) is true. Good.
```

Object class as a special case
The Object class can appear as a special case in inheritance functionality. The compiler bypasses type checking
for assignments to and from variables that are declared to be of type Object. Some classes inherit from the Object class, some classes inherit from another class, and some classes don’t inherit from any class. Although the Dialog class doesn’t inherit from any class, the assignment and call statements in the following code example work. However, if the assignment is bank4 = dlog3; , it will fail at compile time, because the Bank and Dialog classes have no inheritance relationship to each other. The compiler performs only one small validation on assignments to a variable that is declared to be of the Object class. The compiler verifies that the item that is being assigned to the Object variable is an instance of a class. The compiler doesn’t allow an instance of a table buffer to be assigned to the Object variable. Additionally, the compiler doesn’t allow primitive data types, such as int or str, to be assigned to the Object variable.

```java
static void ObjectExample()
{
    Bank bank4;
    Object obj2;
    Dialog dlog3 = new Dialog("Test 4.");
    obj2 = dlog3;  // The assignment does work.
    obj2.run(false);  // The call causes the dialog to appear.
    info("Test 4a is finished.");
}
```

Tables
All tables inherit directly from the Common system table, unless they explicitly inherit from a different table. The Common table can’t be instantiated. It doesn’t exist in the underlying physical database. The Common table inherits from the xRecord class, but in a special way that isn’t appropriate for the is keyword or the as keyword. When the as keyword is used to perform an invalid downcast among tables, the target variable references an unusable non-null entity. Any attempt to de-reference the target variable will cause an error that stops the program.

The is and as keywords and extended data types
Each extended data type has an Extends property. The style of inheritance that this property controls differs from the style of inheritance that the is and as keywords are designed for.

Relational operators
The following table lists the relational operators that can be used in X++. Most of the operators are binary and take two operands. However, the not ( ! ) operator is unary and takes only one operand. Syntax for binary operators: expression1 relationalOperator expression2 Syntax for unary operators: relationalOperator expression1

<table>
<thead>
<tr>
<th>OPERATOR</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>like</td>
<td>The like relational operator returns true if expression1 is like expression2.</td>
</tr>
<tr>
<td>==</td>
<td>The equal relational operator returns true if both expressions are equal.</td>
</tr>
<tr>
<td>&gt;=</td>
<td>The greater than or equal to relational operator returns true if expression1 is greater than or equal to expression2.</td>
</tr>
<tr>
<td>&lt;=</td>
<td>The less than or equal to relational operator returns true if expression1 is less than or equal to expression2.</td>
</tr>
</tbody>
</table>
The greater than relational operator returns true if expression1 is greater than expression2.

The less than relational operator returns true if expression1 is less than expression2.

The not equal relational operator returns true if expression1 differs from (that is, if it isn’t equal to) expression2.

The and relational operator returns true if both expression1 and expression2 are true.

The or relational operator returns true if expression1 or expression2 is true, or if both are true.

The not or unary relational operator negates the expression. It returns true if the expression is false and false if the expression is true.

The like operator

The like operator can use * as a wildcard character for zero or more characters, and ? as a wildcard character for one character. The maximum length of the operand is 1,000 characters. The like operator is evaluated by the underlying SQL, so the result might differ on different installations. If the expressions that you're comparing contain a file path, you must include four backslashes between each element, as shown in the following example.

```sql
select * from xRefpaths
where xRefPaths.Path like "\\\Classes\\\AddressSelectForm"
```

The equal (=) operator

When you use the equal (=) operator to compare objects, the object references are compared, not the objects themselves. This behavior might cause issues if you compare two objects, one of which is located on the server, and the other of which is located on the client. In these cases, you should use the equal method in the Object class. You can override this method to specify what it means for two objects to be equal. If you don't override the equal method, the comparison is identical to the comparison that is done by the equal (=) operator.

Code examples for relational operators

```plaintext
"Jones" like "Jo?es" // Returns true, because the ? is equal to any single character.
"Fabrikam, Inc." like "Fa*" // Returns true, because the * is equal to zero or more characters.
((42 * 2) == 84) // Returns true, because 42*2 is equal to 84.
today() >= 1\1\1980 // Returns true, because today is later than January 1, 1980.
((11 div 10) >= 1) // Returns true, because 11 div 10 is 1 (therefore, >= 1 is true).
(11 <= 12) // Returns true, because 11 is less than 12.
((11 div 10) > 1) // Returns false, because 11 div 10 is 1.
(11 div 10) < 1) // Returns false, because 11 div 10 is 1.
(11 != 12) // Returns true, because 11 is not equal to 12.
(1 == 1) && (3 > 1) // Returns true, because both expressions are true.
```

Operator precedence

The order that a compound expression is evaluated in can be important. For example, \((x + y / 100)\) gives a
different result, depending on whether the addition or the division is done first. You can use parentheses \((x + y) / 100\) to explicitly tell the compiler how it should evaluate an expression. For example, you can specify \((x + y) / 100\). If you don’t explicitly tell the compiler the order that you want operations to be done in, the order is based on the precedence that is assigned to the operators. For example, the division operator has higher precedence than the addition operator. Therefore, for the expression \(x + y / 100\), the compiler evaluates \(y / 100\) first. In other words, \(x + y / 100\) is equivalent to \(x + (y / 100)\). To make your code easy to read and maintain, be explicit. Use parentheses to indicate which operators should be evaluated first. The following table lists the operators in order of precedence. The higher an operator appears in the table, the higher its precedence. Operators that have higher precedence are evaluated before operators that have lower precedence. Note that the operator precedence of X++ isn’t the same as the operator precedence of other languages, such as C# and Java.

<table>
<thead>
<tr>
<th>OPERATOR GROUPS, IN ORDER OF PRECEDENCE</th>
<th>OPERATORS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unary</td>
<td>- ~ !</td>
</tr>
<tr>
<td>Multiplicative, shift, bitwise AND, bitwise exclusive OR</td>
<td>* / % DIV &lt;&lt; &gt;&gt; &amp; ^</td>
</tr>
<tr>
<td>Additive, bitwise inclusive OR</td>
<td>+ -</td>
</tr>
<tr>
<td>Relational, equality</td>
<td>&lt; &lt;= == != &gt; &gt;= like as is</td>
</tr>
<tr>
<td>Logical (AND, OR)</td>
<td>&amp;&amp;</td>
</tr>
<tr>
<td>Conditional</td>
<td>? :</td>
</tr>
</tbody>
</table>

Operators on the same line have equal precedence. If an expression includes more than one of these operators, it’s evaluated from left to right, unless assignment operators are used. (Assignment operators are evaluated from right to left.) For example, \(\&\&\) (logical AND) and \(||\) (logical OR) have the same precedence, and are evaluated from left to right. Therefore:

- \(0 \&\& 0 || 1\) is equal to \(1\)
- \(1 || 0 \&\& 0\) is equal to \(0\).
This topic describes operator precedence.

The order in which a compound expression is evaluated is important. If you do not explicitly tell the compiler the order that you want operations to be performed in, the order is based on operator precedence. You can use parentheses ( ) to explicitly tell the X++ compiler how you want an expression to be evaluated.

Consider the expression \(x + y / 100\), which gives a different result depending on whether the addition or the division is performed first. Because the division operator has a higher precedence than the addition operator, the compiler evaluates \(y/100\) first. So, \(x + y / 100\) is equivalent to \(x + (y / 100)\). If you add parentheses to make the expression \((x + y)/100\), then \(x + y\) is evaluated first.

To make your code easy to read and maintain, be explicit, and indicate with parentheses which operators should be evaluated first.

### Order of operator precedence

The operators in the following table are listed in precedence order. The higher in the table an operator appears, the higher precedence it has. Operators with higher precedence are evaluated before operators with a lower precedence. The operator precedence of X++ is not the same as other languages, for example C# and Java.

<table>
<thead>
<tr>
<th>OPERATORS IN PRECEDENCE ORDER</th>
<th>SYNTAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>unary operators</td>
<td>- ~ !</td>
</tr>
<tr>
<td>multiplicative, shift, bitwise AND, bitwise exclusive OR</td>
<td>* / % DIV &lt;&lt; &gt;&gt; &amp; ^</td>
</tr>
<tr>
<td>additive, bitwise inclusive OR</td>
<td>+ -</td>
</tr>
<tr>
<td>relational, equality</td>
<td>&lt; &lt;= == != &gt; &gt;= like as is</td>
</tr>
<tr>
<td>logical operators (AND, OR)</td>
<td>&amp;&amp;</td>
</tr>
<tr>
<td>conditional</td>
<td>? :</td>
</tr>
</tbody>
</table>

Operators on the same line in the table have equal precedence. If there are more than one of these operators in an expression, the expression is evaluated from left to right unless assignment operators are used. Assignment operators are evaluated from right to left. For example, \(&&\) (logical AND) and \(||\) (logical OR) have the same precedence and are evaluated from left to right. This means that \(0 && 0 || 1 == 1\), and \(1 || 0 && 0 == 0\).
This topic describes how to create and use classes in X++.

A class is a software construct that defines the data and methods of the instances that are later constructed from that class. The class is an abstraction of an object in the problem domain. The instances that are constructed from the class are known as instances or objects. This topic uses the term instance. The data represents the state of the object, whereas the methods represent the behavior of the object.

Variables contain the data for the class, and are called fields. Every instance that is constructed from the class declaration has its own copy of the variables. These variables are known as instance variables or instance fields. This topic will use the term field in most cases.

Methods define the behavior of a class. They are the sequences of statements that operate on the data (instance fields). By default, methods are declared to operate on the instance fields of the class. These methods are known as instance methods or object methods.

You can declare static methods and static fields, that do not have access to instance fields. These are described in X++ static classes.

### Declare a class

You must use the Add new item dialog in Visual Studio to add a class to your project.

1. In Server Explorer, right-click the project, and then click Add.
2. In the New Item dialog box, select Installed > Dynamics 365 Items > Code in the left navigation. Then select Class, and then enter a name for the class.
3. Click Add.

All classes are public. If you remove the public modifier, the system still treats the class as public. You can specify other modifiers on the class declaration, such as final and extends.

### Fields

Instance fields are protected by default. This means that they can only be accessed in the same class or a derived class. You can modify an instance field declaration by using the private, protected, or public keywords.

The following example shows how to use accessor methods to make the field data public. The field firstName is protected, so accessor (get and set) methods are implemented to allow access to the protected field. The field lastName is public, so code can directly get and set the value of the field.
// This is the class definition.
public class HasAFirstName
{
    str firstName = "";
    public str lastName = "";
    public str getFirstName()
    {
        return firstName;
    }
    public void setFirstName(str newName)
    {
        firstName = newName;
    }
}

// This code creates an instance of the class and gets the fields.
public static void TestLastName()
{
    HasAFirstName hasFirstName = new HasAFirstName();
    hasFirstName.setFirstName("Dion");
    info(hasFirstName.getFirstName());
    hasFirstName.lastName = ("Townes");
    info(hasFirstName.lastName);
}
// The output is "Dion" and "Townes".

Field attributes
You can decorate a field with an attribute, in the same way that attributes can decorate classes and methods. The following example decorates the myField field with the MyAttribute attribute.

class MyClass
{
    [MyAttribute]
    public int myField;
}

One particularly useful attribute is the SysObsolete attribute. If the SysObsolete attribute is applied to a field, then the compiler generates an error or warning on any reference to the field. Whether it's a warning or error depends on the second parameter in the attribute.

class MyClass
{
    [SysObsolete("This field is obsolete.", true)]
    public int myField;
}

Constructors
To create an instance of a class, you must instantiate it by using a constructor.

- You can define only one new method (constructor) in a class.
- If you do not define a constructor, a default constructor with no parameters is created automatically by the compiler.
- You can simulate a default constructor by assigning default values to the parameters in the new method.

The following example defines a parameterless constructor in the Point class.
class Point
{

    // Instance fields that are public. In practice, you would probably make this protected or private.
    // and create accessor methods.
    public real x = 0.0;
    public real y = 0.0;

    void new() {
    }
}

Following is information about how to create a clean inheritance model and minimize problems when code is upgraded:

- Each class must have a single public construction method unless the class is abstract. If no initialization is required, use a static construct method. Otherwise, use a static new method (the default constructor for the class should be protected).
- Each class should have at least one static construct method.
- Each class should have at least one static new method.
- Each class should have a new method (the default constructor). This method should be protected.
- Create accessor methods to get and set class fields.
- Create init methods to carry out any specialized initialization tasks that should be carried out after instantiation.

Create other objects in a constructor

A class constructor can instantiate other objects in addition to creating an instance of the class. For example, the following code declares a Rectangle class that uses two Point objects to define its bounds. In this case, the Point class has a constructor that has two real parameters.
class Point
{
    // Instance fields that are public. In practice, you would probably make this protected or private.
    // and create accessor methods.
    public real x = 0.0;
    public real y = 0.0;

    // Constructor to initialize to a specific or default value
    void new(real _x = 10, real _y = 10)
    {
        x = _x;
        y = _y;
    }
}

class Rectangle
{
    public Point lowerLeft;
    public Point upperRight;

    void new(real _topLeftX = 0.0, real _topLeftY = 0.0, real _bottomRightX = 1.0, real _bottomRightY = 1.0)
    {
        lowerLeft = new Point(_topLeftX, _topLeftY);
        upperRight = new Point(_bottomRightX, _bottomRightY);
    }
}

// This code creates two instances of the Rectangle class.
Rectangle defaultRectangle = new Rectangle();
info(any2Str(defaultRectangle.lowerLeft.x));
info(any2Str(defaultRectangle.lowerLeft.y));
// Output is "0.0" and "0.0".

Rectangle customRectangle = new Rectangle(1.0, 1.0, 2.0, 2.0);
info(any2Str(customRectangle.lowerLeft.x));
info(any2Str(customRectangle.lowerLeft.y));
// Output is "1.0" and "1.0".

Create an instance of an object

The constructor, `new`, returns a new instance of the class. The following code example creates two instances of the Point class.

```java
// Declare a field to refer to a Point instance.
Point myPoint;

// Create an instance of the Point class.
myPoint = new Point();

// Declare and instantiate a Point instance.
Point ap = new Point();
```

Destructors

You use a destructor to explicitly destroy a class instance. Instances are automatically destroyed when there are no references to them. However, you can destroy objects explicitly in the following ways:

- Use the `finalize` method.
- Set the reference variable to `null`. 
Use the finalize method

Use the `finalize` method to explicitly destroy an object. There are no implicit calls to the `finalize` method. You must call the method to run the statements in it. In the `finalize` method, you should also put any clean-up code that is required. For example, if your class uses a dynamic-link library (DLL) module, you can use the `finalize` method to release the DLL when you no longer require it. Use the `finalize` method carefully. It will destroy an object even if there are references to it.

The following example shows the basic structure for a call to the `finalize` method.

```java
// From any method in a class.
if (condition)
{
    // Removes object from memory.
    this.finalize();
}
```

Set reference variable to null

Set the reference variable to `null` to terminate an object. This approach destroys an object only if no other variables point to that object. You should verify that other code isn’t using the variable. The following example creates a reference variable and then sets it to `null`.

```java
Point myPoint = new Point();
myPoint = null;
```

Methods

Instance methods

Instance methods are embedded in each instance that is created from the class. You must instantiate the object before you can use the method. The following code shows how to define an instance method and call it from an instance.

```java
class Square
{
    int side = 0;

    void new(int _side = 1) {
        side = _side;
    }

    int getArea() {
        return side * side;
    }
}

// This code creates an instance of Square and calls getArea.
Square square = new Square(15);
int area = square.getArea();
info(int2Str(area));
// Output is "225".
```

Static methods

Static methods, which are also known as class methods, belong to a class and are created by using the keyword `static`. You don’t have to instantiate an object before you use static methods. Static methods are often used to work with data that is stored in tables. Member fields can’t be accessed from a static method.
You use the following syntax to call static methods.

```
ClassName::methodName();
```

If you convert an instance method to a static method, you must restart the client. Otherwise, the compiler doesn't detect the change. After you've converted an instance method to a static method, you can no longer call the method from the instance of the class. Instead, you must call the method from the class itself. For more information about static methods, see X++ static classes.

**main methods**

A main method is a class method that is run directly from a menu option. The method should only create an instance of the object and then call the required member methods. The _args parameter lets you transfer data to the method.

```
static void main (Args _args)
{
   // Your code here.
}
```

**Declaration of methods**

Method declarations consist of a header and a body. The method header declares the method's name and return type), the method modifiers, and parameters. (The return type might be void.) The method body consists of fields declarations, method declarations, and statements.

**Return type**

A return type is required for each method. If a method doesn't return anything, use the void keyword as the return type.

The following example shows two methods. One method has a return type, but the other method doesn't have a return type.

```
void methodNameNoReturnValue()
{
   // Your code here.
}

// If a method returns something, you must specify the return type and include a return statement.
int methodNameIntegerReturnValue()
{
   return 1;
}
```

**Syntax**

Method declaration =  \[ Heading \]  Body  \[ Heading \]  =  \[ Modifiers \]  ReturnType  MethodName( ParameterList)

Modifiers = [client] [server] [edit | display | public | protected | private] [static | abstract | final ]

ReturnType = Datatype | void | anytype

MethodName = Identifier

ParameterList = [ Parameter { , Parameter} ]

Parameter = Datatype VariableIdentifier [ = Expression]

Body = [ { VariableDeclarations} { EmbeddedFunctionDeclarations} [ Statements] ]
Example of a method that doesn't have a return type

```java
void update ()
{
    // Field declared and initialized
    CustTable this_Orig = this.orig();

    // First statement in body (begin transaction)
    ttsBegin;
    this.setNameAlias();
    // Calls super's implementation of update
    super();
    this.setAccountOnVend(this_Orig);
    if (this_Orig.custGroup != this.custGroup)
    ForecastSales::setCustGroupId(
        this.accountNum,
        this_Orig.custGroup,
        this.custGroup);
    // Commits transaction
    ttsCommit;
}
```

Example of a method that has parameters

In the following example, the `checkAccountBlocked` method returns a Boolean value and acts on the `amountCur` parameter.

```java
boolean checkAccountBlocked(AmountCur amountCur)
{
    if (this.blocked == CustVendorBlocked::All
 || (this.blocked == CustVendorBlocked::Invoice
    && amountCur > 0 ))
    return checkFailed(strFmt("@SYS7987",this.accountNum));
    return true;
}
```

Method modifiers

Several modifiers can be applied to method declarations. Some of the modifiers can be combined (for example, `final static`). Here are the method modifier keywords:

- **abstract**: The method is declared but isn't implemented in a parent class. The method must be overridden in subclasses. If you try to create an object from a subclass where one or more abstract methods that belong to the parent class haven't been overridden, you receive a compiler error.

  Classes can also be abstract. Sometimes, a class should not be instantiated even though it represents an abstract concept. Only subclasses should be instantiated. Base classes of this type can be declared as `abstract`. For example, you want to model the concept of an account. Accounts are abstract, because only derived classes (ledger accounts and so on) exist in the real world. This example describes a clear case where you should declare the `Account` class as `abstract`.

- **display**: The method's return value should be shown on a page or a report. The value can't be modified on the page or report. Typically, the return value is a calculated value, such as a sum.

- **edit**: The method's return type should be used to provide information for a field that is used on a page. The value in the field can be modified.
• **final**: The method can't be overridden in any class that derives from its class.

• **public**: Methods that are declared as public can be accessed anywhere that the class is accessible, and they can be overridden by subclasses. Methods that have no access modifier are implicitly public.

• **protected**: Methods that are declared as protected can be called only from methods in the class and in subclasses that extend the class where the method is declared.

• **private**: Methods that are declared as private can be called only from methods in the class where the private method is declared.

• **static**: The method is a class method and doesn't act on an instance. Static methods can't refer to instance fields. They aren't invoked on an instance of the class. Instead, they are invoked by using the class name (for example, `MyClass::aStaticProcedure()`).

### Methods that have modifiers

The following examples show only the method headers.

```java
// A method that cannot be overridden
final int dontAlterMe()

// A static method
static void noChange()

// A display method that returns an integer
display int value()
```

### Method access control

You use the accessor keywords **public**, **protected**, and **private** to control whether the methods in other classes can call the methods on your class. The accessor keywords on methods also interact with the rules for class inheritance. Here are the accessor keywords that you use with methods:

- **public**: Methods that are declared as public can be called from anywhere that the class is accessible. In addition, a public method can be overridden by a subclass, unless the method is declared as **final**.

- **protected**: Methods that are declared as protected can be called only from the following methods:
  - Methods in the class.
  - Methods in a subclass of the class that contains the protected method. Methods that are protected can be overridden in subclasses.

- **private**: Methods that are declared as private can be called only from methods in the class where the private method is declared. No private method can be overridden in a subclass. By default, when you create a new method, the private accessor keyword appears in the code editor. For maximum security, **private** is the most conservative default accessor keyword.

### Static and instance methods

The accessor keywords on methods never restrict calls between two methods that are in the same class, regardless of which method is static or non-static. In a static method, calls to the **new** constructor method are valid even if the **new** constructor method is decorated with the **private** modifier. The syntax for these calls requires that the **new** keyword be used. The code in a static method must construct an instance object of its own class before it can call any instance methods on the class.

### Increasing access during overrides

When a method is overridden in a subclass, the overriding method must be at least as accessible as the overridden method. For example, the following compiler rules apply when a protected method is overridden in a subclass:
- A public method in a superclass can be overridden only by a public method in the subclass.
- In a subclass, a public or protected method can override a protected method of the superclass.
- In a subclass, a private method can't override a protected method of the superclass.

Optional parameters

Parameters can be initialized in the method declaration. In this case, the parameter becomes an optional parameter. If no value is supplied in the method call, the default value is used. All required parameters must be listed before the first optional parameter. The following examples show how to create and call a method that has optional parameters. The example of the AddThreeInts method shows that you can't skip default parameters when you call a method.

Examples of optional parameters

The following code example shows a class with a default parameter.

```java
// This is an example of a function being used as the default.
class Person
{
    date birthDate;

    // The constructor that takes a date type as a parameter.
    // That value is assigned to the field member birthDate.
    void new(date _date)
    {
        birthDate = _date;
    }

    // The CalculateAgeAsOfDate method references the birthDate field and has an
    // optional parameter. In this example, the default value is the
    // return value of a function.
    public real CalculateAgeAsOfDate(date _calcToDate =
        DateTimeUtil::getToday(DateTimeUtil::getUserPreferredTimeZone()) )
    {
        return (_calcToDate - birthDate) / 365;
    }

    public static void callPerson()
    {
        Person person = new Person(13\5\2010);

        // Optional parameter's default is used.
        Info(strFmt('Age in years today is %1 years',
            real2int(person.CalculateAgeAsOfDate())));

        // January 2, 2044 is the parameter value for _date.
        Info(strFmt('Age in years on %1 is %2 years',
            2\1\2044,
            real2int(person.CalculateAgeAsOfDate(2\1\2044))));
    }
}
```

This is an example of how you cannot skip to a second optional parameter. The AddThreeInts method has two optional parameters. The callAdditions method calls the AddThreeInts method. The commented out code tries to override only the _i3 default value, but the compiler requires that all prior optional parameters also be overridden in the call.
```java
class Additions {
    public static int AddThreeInts(int _i1, int _i2 = 2, int _i3 = 3) {
        return _i1 + _i2 + _i3;
    }

    public static void callAdditions() {
        // The next statement does not compile, because it skips the _i2 parameter.
        // info(int2Str(Additions::AddThreeInts(1, , 99)));

        // You must specify both optional parameters.
        info(int2Str(Additions::AddThreeInts(1, 2, 99)));
    }
}
```

### Accessor methods

Class fields are protected by default. By hiding details of the internal implementation of a class, you can change the implementation of the class later without breaking any code that uses that class. To access the data from reference fields, you must create accessor methods. The following example defines a `Point` class that uses accessor methods to access the fields `x` and `y`.

```java
class Point {
    // Instance fields
    real x;
    real y;

    // Constructor to initialize to a specific or default value
    void new(real _x = 10, real _y = 10) {
        x = _x;
        y = _y;
    }

    // Accessor methods
    void setX(real _x) {
        x = _x;
    }

    void setY(real _y) {
        y = _y;
    }

    real getX() {
        return x;
    }

    real getY() {
        return y;
    }
}
```

These method declarations show how the `Point` class provides access to its fields from the outside world. Other objects can manipulate the instance fields of `Point` objects by using the accessor methods.
Point myPoint = new Point();
// Set the x fields using the accessor method.
myPoint.setX(4.0);
// Get the x fields using the accessor method.
info(any2Str(myPoint.getX()));

Parameters

All methods have their own *scope*. A method can take one or more parameters. Within the scope of the method, these parameters are treated as local variables and are initialized with a value from the parameter in the method call. All parameters are passed by value, which means that you can't change the value of the original variable. You can change only the local variable in the method. This local variable is a copy of the original variable.

Scope of variables in methods

A scope defines the area in which an item can be accessed. Variables that are defined in a class are available to the methods within that class. Variables in methods can be accessed only within the current block.

Local functions

You can declare functions inside a method. These are called local functions. While possible, it is not a best practice. Instead, you should add private methods to the class.

- The declarations of local functions must physically precede any non-declaration statements in the method.
- You can declare more than one local function in your method. However, all local functions must be declared in an uninterrupted series, and the set must be terminated by one semicolon (;).
- Code that is inside the local function can access variables that are declared in the method that contains the local function.
- Code that is outside the local function can't access variables that are declared in the local function.
- A local function can be called only by code in the same method where the local function is declared.
- A local function should never call itself. Such recursion can prevent successful compilation.

The following example shows valid declarations of two local functions, `localFunctionA` and `localFunctionB`. Calls to the local functions occur after the function declarations in the example, as is required.
static void StaticFunction()
{
    int number = 654;

    void localFunctionA(int _iNum) // The local function.
    {
        str innerString = "String in localFunctionA";
        str output = strFmt("localFunctionA: %1 , %2 , %3", _iNum, innerString, number);
        info(output);
    }

    void localFunctionB()
    {
        info("Printing from inside localFunctionB.");
    }

    localFunctionA(55);
    localFunctionB();
    // Next info statement would fail to compile,
    // because innerString is restricted to the
    // scope of the local function in which it is declared.
    // print innerString;
}

// When called, the output is:
// localFunctionA: 55 , String in localFunctionA , 654
// Printing from inside localFunctionB.

Extension methods

The extension method feature lets you add extension methods to a target class by writing the methods in a separate extension class. The following rules apply:

- The extension class must be static.
- The name of the extension class must end with the ten-character suffix _Extension. However, there's no restriction on the part of the name that precedes the suffix.
- Every extension method in the extension class must be declared as public static.
- The first parameter in every extension method is the type that the extension method extends. However, when the extension method is called, the caller must not pass in anything for the first parameter. Instead, the system automatically passes in the required object for the first parameter.

It's perfectly valid to have private or protected static methods in an extension class. These are typically used for implementation details and are not exposed as extensions. The example below illustrates an extension class holding a few extension methods:
public static class AtlInventLocation_Extension
{
    public static InventLocation refillEnabled(
        InventLocation _warehouse,
        boolean _isRefillEnabled = true)
    {
        _warehouse.ReqRefill = _isRefillEnabled;
        return _warehouse;
    }

    public static InventLocation save(InventLocation _warehouse)
    {
        _warehouse.write();
        return _warehouse;
    }
}

Reasons to use extension methods
The extension method technique doesn’t affect the source code of the class it extends. Therefore, the addition to
the class can be done without over-layering. Upgrades to the target class are never affected by any existing
extension methods. However, if an upgrade to the target class adds a method that has the same name as your
extension method, your extension method becomes unreachable through objects of the target class. Extension
methods are easy to use. The extension method technique uses the same dot-delimited syntax that you routinely
use the call regular instance methods. Extension methods can access all public artifacts of the target class, but
they can’t access things that are protected or private. In this way, extension methods can be seen as a kind of
syntactic sugar.

Where can extension methods be applied
The target of an extension method must be one of the following application object types:

- Class
- Table
- View
- Map

Regardless of the target type, an extension class is used to add extension methods to the type. For example, an
extension table is not used to add methods to a table, and there’s no such thing as an extension table.

The this keyword
The this keyword is a reference to the instance of the class or table where the this keyword is used. The this
reference is never required, but it can clarify your code and enhances the behavior of IntelliSense in the code
editor. All calls to instance methods must be qualified by either the this reference or a variable. The this
reference can be used to qualify the following information:

- The names of other instance (non-static) methods in the same class where the this reference is used. Here is
  an example: boolColorChanged = this.colorItOrange();
- The names of methods that are inherited by the this object.
- The names of fields on the table that contains the method that the this keyword is used in.

The this reference can’t be used in the following ways:

- It can’t qualify the names of member variables that are declared in the classDeclaration code.
- It can’t be used in a static method.
- It can’t qualify the names of static methods of the class or table.
Nested classes

Classes can be nested in X++ source code. Nested classes are available only inside forms (such as a class that extends FormRun) to represent controls, data sources, or data fields.

Jobs

There is no concept of an X++ job from preview versions (AX2102 and earlier). To quickly and easily run an X++ method, add a `static Main` method to a class, and then set the class as the startup object form for the project in Microsoft Visual Studio. When the project is run, the `Main` method will be run.

Call stack limitation

The depth of the call stack is limited to 100.
This topic describes inheritance in X++, including how to create a subclass and override a method.

### Creating a subclass

**Subclasses** are classes that extend or inherit from other classes. A class can extend only one other class. Multiple inheritance isn't supported. If you extend a class, the subclass inherits all the methods and variables in the parent class (the *superclass*). Subclasses let you reuse existing code for a more specific purpose. Therefore, they help save you time during design, development, and testing. To customize the behavior of a superclass, override the methods in a subclass. A superclass is often known as a *base class*, and a subclass is often known as a *derived class*.

**Subclass example**

The following example first creates a class that is named `Point`. It then extends the `Point` class to create a new class that is named `ThreePoint`.

```plaintext
class Point
{
    // Instance fields.
    real x;
    real y;

    // Constructor to initialize fields x and y.
    void new(real _x, real _y)
    {
        x = _x;
        y = _y;
    }
}

class ThreePoint extends Point
{
    // Additional instance fields z. Fields x and y are inherited.
    real z;

    // Constructor is overridden to initialize z.
    void new(real _x, real _y, real _z)
    {
        // Initialize the fields.
        super(_x, _y);
        z = _z;
    }
}
```

### Preventing class inheritance

You can prevent classes from being inherited by using the `final` modifier.

```plaintext
public final class Attribute
{
    int objectField;
}
```
Overriding a method

The methods in a class are inherited by any class that extends the class. To change the functionality of an inherited method, you create a method in the subclass, and then give that method the same name and parameters as the method in the superclass. This process is known as *overriding* the method. In the following example, `ColorAttribute` is a subclass of `Attribute` and therefore inherits the `methodAttr` method. However, because `ColorAttribute` defines a method that has the same name and the same number of arguments, the method in the superclass is overridden.

When you instantiate the subclass, you can assign the reference to either a variable of the superclass type or the subclass type. Regardless of the type of the variable, the overridden method is called.

In the following code example, the subclass overrides the `write` method. Two variables, both of type `Point` are created. One is assigned a `Point` object, the other is assigned a `ThreePoint` object. When the `write` method is called on the `ThreePoint` object, the `ThreePoint` version of the method is called.
class Point
{
    // Instance fields.
    real x;
    real y;

    // Constructor to initialize fields x and y.
    void new(real _x, real _y)
    {
        x = _x;
        y = _y;
    }

    void write()
    {
        info("(" + any2Str(x) + ", " + any2Str(y) + ")");
    }
}

class ThreePoint extends Point
{
    // Additional instance fields z. Fields x and y are inherited.
    real z;

    // Constructor is overridden to initialize z.
    void new(real _x, real _y, real _z)
    {
        // Initialize the fields.
        super(_x, _y);
        z = _z;
    }

    void write()
    {
        info("(" + any2Str(x) + ", " + any2Str(y) + ", " + any2Str(z) + ")");
    }
}

// Code that creates Point objects and calls the write method.
Point point2 = new Point(1.0, 2.0);
Point point3 = new ThreePoint(3.0, 4.0, 5.0);
point2.write(); // Output is "(1.0, 2.0)".
point3.write(); // Output is "(3.0, 4.0, 5.0)".

Preventing method overrides
Static methods can’t be overridden, because they exist per class. To protect other sensitive methods, or core
methods, from being overridden, use the final modifier. In the following example, because methodAtt is
declared as final, it can’t be overridden in any class that extends Attribute. You should not specify new or
finalize methods as final.

The following example shows how to use the final keyword.
public class Attribute
{
    int objectVariable;

    final void methodAtt()
    {
        //Some statements
    }
}

**Overriding vs. overloading**

Overriding occurs when the superclass’s implementation of a method is changed by the subclass’s implementation of that method, but the signatures of both methods are the same.

By contrast, *overloading* occurs when more than one method has the same name, but the methods have different signatures (return types, parameter lists, or both). X++ supports overriding, but it doesn’t support overloading.
This topic describes static class members in X++. In general, static methods are intended for these cases:

- The method has no reason to access the member variables that are declared in the class.
- The method has no reason to call any instance (non-static) methods of the class.

You declare static class members by using the `static` keyword. The `static` keyword instructs the system to create only one instance of the method, regardless of the number of instances of the class there are. This one instance is used throughout your session.

**Static methods**

This section describes a scenario where a software key type is used to help prevent piracy. Each instance of a software key can have its own unique value. Because all software keys must conform to the rules of software key design, the logic that tests for software key conformance is the same for all software keys. Therefore, the method that contains the conformance validation logic should be static.

Here is an example of a method that is declared by using the `static` keyword.

```plaintext
class SoftwareKey
{
    public static boolean validateSoftwareKey(String _softwareKeyString)
    {
        // Your code here.
        return false;
    }
}
```

In the following example, you don't have to construct an instance of the `SoftwareKey` class before you call a static method on the class. When you want to call the static `validateSoftwareKey` method, the syntax starts with the name of the class that contains the method. A pair of colons (::) is used to connect the class name to the static method name.

```plaintext
boolean yourBool = SoftwareKey::validateSoftwareKey(yourSoftwareKeyString);
```

**Static fields**

Static fields are variables that are declared by using the `static` keyword. Conceptually, they apply to the class, not to instances of the class.

**Static constructors**

A static constructor is guaranteed to run before any static or instance calls are made to the class. The execution of the static constructor is relative to the user’s session. The static constructor has the following syntax.

```plaintext
static void TypeNew()
```

You never explicitly call the static constructor. The compiler will generate code to make sure that the constructor is called exactly one time before any other method on the class. A static constructor is used to initialize any static data or perform a particular action that must be performed only one time. No parameters can be provided for
the static constructor, and it must be marked as `static`.

The following code example shows how to create a singleton instance by using a static constructor.

```java
public class Singleton
{
    private static Singleton instance;

    private void new()
    {
    }

    static void TypeNew()
    {
        instance = new Singleton();
    }

    public static Singleton Instance()
    {
        return Singleton::instance;
    }
}
```

The singleton guarantees that only one instance of the class will ever be called. The following example shows how to instantiate the singleton.

```java
Singleton i = Singleton::Instance();
```

**Static methods**

Static methods, which are also known as *class methods*, belong to a class and are created by using the keyword `static`. You don’t have to instantiate an object before you use static methods. Static methods are often used to work with data that is stored in tables. Member variables can’t be used in a static method. You use the following syntax to call static methods.

```java
ClassName::methodName();
```

**Static and instance methods**

The accessor keywords on methods never restrict calls between two methods that are in the same class, regardless of which method is static or non-static. In a static method, calls to the `new` constructor method are valid even if the `new` constructor method is decorated with the `private` modifier. The syntax for these calls requires that the `new` keyword be used. The code in a static method must construct an instance object of its own class before it can call any instance methods on the class.
An interface specifies a set of public instance methods. An interface defines and enforces similarities between unrelated classes without having to derive one class from the other.

All interfaces are public, even if you don't explicitly add the public keyword in front of the interface keyword in interface declaration. The methods on an interface are also public. Explicit inclusion of the keyword public is optional.

To create an interface, follow these steps.

1. In Server Explorer, right-click the project, and then click Add.
2. In the New Item dialog box, select Interface, and then enter a name for the interface.
3. Click Add.

When you add the implements keyword on a class declaration, the class must declare and define the methods that are specified by the interface. A class declaration can implement multiple interfaces. List the interfaces after the single occurrence of the implements keyword, and separate the interface names by using commas.

All interface methods that a class implements must be explicitly declared as public. A class that implements an interface must also be declared as public. An interface can extend another interface by using the extends keyword, however, an interface can't extend more than one interface.

It is customary to preface the name of an interface with I.

### Interface example

In the following code example, the Automobile class implements the IDrivable interface. The is keyword tests whether a class implements an interface.
interface IDrivable
{
    int getSpeed()
    {
    }

    void setSpeed(int newSpeed)
    {
    }
}

class Automobile implements IDrivable
{
    int speed;

    public int getSpeed()
    {
        return speed;
    }

    public void setSpeed(int newSpeed)
    {
        speed = newSpeed;
    }
}

class UseAnAutomobile
{
    void DriveAutomobile()
    {
        IDrivable drivable;
        Automobile myAutomobile = new Automobile();
        str temp;
        myAutomobile = new Automobile();

        if (myAutomobile is IDrivable)
        {
            drivable = myAutomobile;
            drivable.setSpeed(42);
            temp = int2str(drivable.getSpeed());
        }
        else
        {
            temp = "Instance is not an IDrivable.";
        }
        info(temp);
    }
}
This topic describes the library of classes in X++.

There are two kinds of classes: application classes and system classes.

- **Application classes** – These classes are implemented in X++. They are available in the Code > Classes node in Application Explorer.
- **System classes** – These classes are sometimes known as kernel classes. They are listed under the System Documentation > Classes node in Application Explorer. The source code for these classes isn't available. For a list of the system classes, see API, class, and table reference.

**Typical structure of an application class**

The following code block types are standard for application classes:

- **class and variable declarations**: The class declaration contains modifiers, such as public, private, and extends.
- **variable declarations**: These are the field members for objects that are constructed from the class. When you type the keyword this on a class instance variable, IntelliSense can show a list of the members.
- **new method**: This method creates an instance of the class. The constructor can be called only by using the new keyword. Derived classes can call the new method of their constructor by calling the super method reference. For more information, see X++ inheritance.
- **finalize method**: This method finalizes an instance of the class. This method is the destructor method. However, it's a destructor by convention only. The system doesn't automatically call the finalize method during garbage collection.

Additional methods for a class have the following types:

- Instance methods
- Static methods
- Main methods

Methods can be created on many kinds of items. Here are some examples:

- Classes
- Maps
- Views
- Data Sets
- Forms
- Queries

**Substituting application classes for system classes**

You should use the substitute application classes instead of the system classes that they extend.

In Application Explorer, under System Documentation > Classes, several kernel or system classes have names that begin with a lowercase x. These classes are known as x-system classes. Examples of these system classes are xApplication and xVersionControl. Some of these classes are extended by application classes. For example, the Application class extends the xApplication system class.
The classes that derive from x-system classes are known as substitute application classes. In Application Explorer, under the Classes node, the icon next to the substitute application classes differs from the standard icon.

x-system classes

Some of the substitute application classes are associated with a special global variable that represents an instance of the class. For example, the appl variable references a pre-instantiated object from the Application class. The advantage of the appl variable is that the system maintains the object throughout the scope of your session. Your code would be less efficient if it repeatedly used the new Application() syntax to obtain an instance of the Application class. You should not use the xApplication system class. Instead, use the Application substitute application class.

You can reference the static members of the Application class by using the following standard syntax: Application::checkForNewBatchJobs(). However, to reference the instance members of the Application class, you should use that class’s appl variable, if it exists. This pattern applies to most of the x-system classes. The Session substitute application class is one exception, because there is no special global variable for Session.

The following table lists the x-system classes that have a corresponding substitute application class. The special global variables are also shown for those classes that have one.

<table>
<thead>
<tr>
<th>APPLICATION CLASS</th>
<th>X-SYSTEM CLASS</th>
<th>GLOBAL VARIABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Args</td>
<td>xArgs</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Application</td>
<td>xApplication</td>
<td>appl</td>
</tr>
<tr>
<td>ClassFactory</td>
<td>xClassFactory</td>
<td>classFactory</td>
</tr>
<tr>
<td>Company</td>
<td>xCompany</td>
<td>appl.company</td>
</tr>
<tr>
<td>Global</td>
<td>xGlobal</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Info</td>
<td>xInfo</td>
<td>Infolog</td>
</tr>
<tr>
<td>MenuFunction</td>
<td>xMenuFunction</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Session</td>
<td>xSession</td>
<td>Not applicable</td>
</tr>
<tr>
<td>VersionControl</td>
<td>xVersionControl</td>
<td>versionControl</td>
</tr>
</tbody>
</table>

Example of x-system classes

The following example shows the syntax for using several special variables that reference instances of the substitute application classes.
TreeNode treeNode;
Args args;
FormRun formRun;

// appl variable
info(appl.buildNo());

// company variable
appl.company().reloadRights();

// infolog variable
treeNode = infolog.findNode("\forms\custTable");
info(treeNode.AOTgetProperty("Name")); // Output is "CustTable".

// classFactory variable
args = new Args(formstr(Batch));
formRun = classFactory.formRunClass(args);
formRun.init();
formRun.run();
formRun.detach();
info("Method is ending. This is a message in the Infolog."); // Output is "Method is ending. This is a message in the Infolog."

Batch processing classes

You implement classes by using the batch processing system, and by extending the RunBase and RunBaseBatch classes. To remove the Recurrence button from the Batch processing dialog box, you use the Args::parmEnum method. We recommend that you designate a class to run as a server-bound batch method. Server-bound batch methods are more secure than batch methods that aren't server-bound for the following reasons:

- The method is run by using the permissions of the user who submitted the method.
- The method can use only specific Info and Global class methods to interact with the client that is processing it. This restriction limits interaction with the client.

**Enable a class to run as a server-bound batch method**

1. Create a class that extends the RunBaseBatch class.

2. Override the RunBaseBatch.runsImpersonated method to return a value of true, as shown in the following example.

   ```java
   public boolean runsImpersonated() {
      return true;
   }
   ```

3. Confirm that the class calls only the following Info and Global class methods:
   - add
   - Info.copy
   - Info.cut
   - Info.import
   - Info.export
   - Info.line
   - Info.num
   - Global::error
Removing the Recurrence button from the batch processing dialog box

When you implement a class by using the batch processing system, you can remove the Recurrence button by calling the Args.parmEnum method and passing the NoYes::Yes system enumeration value. The NoYes system enumeration determines whether the Recurrence button is removed from the dialog box. The default value is NoYes::No.

In the following example, the InventTransferMultiShip class is implemented. The BatchDialog::main method creates the Batch processing dialog box.

```java
static void noRecurrenceButton(Args _args)
{
    Args a;
    InventTransferMultiShip inventTransferMultiShip;
    a = new Args();
    inventTransferMultiShip = InventTransferMultiShip::construct();
    a.caller(inventTransferMultiShip);
    a.parmEnum(NoYes::Yes);
    BatchDialog::main(a);
}
```

Image manipulation classes

Two system classes let you to manipulate graphics and icons: Image and Imagelist.

- **Image** – This class lets you load, save, and manipulate individual images. For example, you can capture a screen and save it as an image, crop or rotate an image, or manipulate the color depth.
- **Imagelist** – This class lets you work with a set of images that have common properties, such as the size and transparency color. You can view the image lists that are used in the ImageListAppl application classes.

Query object model

The query object model contains classes that are used to define and run a query. The query objects are used to define the query data source, the fields that are returned, record ranges, and relations to child data sources. The query classes are more visible when you create a dynamic query in code, but they are also used behind the scenes when you create a static query in Application Explorer.

The following table describes the classes in the query object model.

<table>
<thead>
<tr>
<th>SYSTEM CLASS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>QueryRun</td>
<td>This class runs the query and fetches the data.</td>
</tr>
<tr>
<td>Query</td>
<td>This class holds some properties, and has one or more related data sources. It's the top level of the query definition.</td>
</tr>
<tr>
<td>QueryBuildDataSource</td>
<td>This class defines access to a single data source in the query. If there is more than one data source at the same level in a query, separate SQL statements are produced and are run sequentially. If one data source is a child of another data source, a join is created between the two data sources.</td>
</tr>
<tr>
<td>SYSTEM CLASS</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>QueryBuildFieldList</td>
<td>This class defines the fields that are returned from the database. By default, the field list is dynamic, and all fields are returned from the data source table, map, or view. Each data source has only one QueryBuildFieldList object. This object contains information about all selected fields. You can specify aggregate functions, such as SUM, COUNT, and AVG, on the field list object.</td>
</tr>
<tr>
<td>QueryBuildRange</td>
<td>This class defines a subset of records that is returned, based on a single field. A range is translated into a WHERE clause in the query SQL statement. If more than one field is used to limit the query (WHERE clause), the data source will contain more than one range.</td>
</tr>
<tr>
<td>QueryBuildDynalink</td>
<td>This class contains information about a relation (limitation) to an external record. When the query is run, this information is converted to additional entries in the WHERE clause of the query SQL statement. This class can exist only on the parent data source of a query. The child data source will then contain one or more DLLs to the parent data source. The function is used even if the two data sources are put in two different forms but are still synchronized.</td>
</tr>
<tr>
<td>QueryBuildLink</td>
<td>This class specifies the relation between the two data sources in the join. This class can exist only on a child data source.</td>
</tr>
</tbody>
</table>

You can also use the SysDa API to query data.

**System classes overview**

The source for system classes isn't available. A system class can have the following characteristics:

- Static methods (or class methods)
- Dynamic methods
- Properties – These properties are member functions that are used to set properties. An example is LeftMargin.

You can't override system class methods. It isn't our intention that you will use the system classes to design your application objects from scratch. Instead, use them to extend or modify the default functionality in Application Explorer. For example, you can dynamically add extra information to an existing report. Alternatively, you can change the options that are available on a page, based on the user's selection on a previous page.

**Collection classes**

The *collection classes* let you create lists, sets, structs, maps, and arrays.

**Application object classes**

These system classes hold functions that are activated whenever you use Application Explorer to create your application. For example, the system uses the FormDesign class when you define the layout of your form in the Designs node in Application Explorer. These classes also let you to create and modify application objects.

**Integration classes**

The integration with the environment is typically implemented by classes. Here are some examples of the classes in this category:
- **COM** – The call of methods on COM objects.
- **DLL** – The call of Microsoft Windows DLL functions.
- **IO** – Read and write external files.
- **ODBCConnection** – An Open Database Connectivity (ODBC) interface to a foreign database.
This topic describes event terminology and keywords in X++.

You can use the event design pattern to make your code more modular and reusable. The term *event* is a metaphor that explains how delegates are used. When something important occurs during a program run, other modules might have to process the occurrence. These important occurrences are known as *events*. When an event occurs, the program tells its notifier for the event that the notifier must send notifications about the event. A notification must be sent to all the event handlers that are subscribers of the notifier. When the program tells its notifier to send the notifications, we call that process *raising* an event.

A delegate can be defined in a table, form, or query, and not just in a class.

The following table shows the terms that are used to describe the event metaphor.

<table>
<thead>
<tr>
<th>TERM</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event</td>
<td>An important occurrence in a program module where additional modules must process the occurrence.</td>
</tr>
<tr>
<td>Notifier</td>
<td>The program element that sends information about the event to all the event handlers that are subscribed to the notifier.</td>
</tr>
<tr>
<td>Subscriber</td>
<td>The program functions or methods that are subscribed to an event notifier.</td>
</tr>
<tr>
<td>Event handler</td>
<td>The methods that subscribe to an event notifier. Only the appropriate kind of methods can be event handlers.</td>
</tr>
</tbody>
</table>

**Keywords that are used for programming that uses delegates**

The following table shows the keywords that describe the use of delegates.

<table>
<thead>
<tr>
<th>KEYWORD OR TERM</th>
<th>CODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>delegate</td>
<td>delegate myDelegate(str information) {}</td>
<td>The code shows what the delegate looks like in the code editor. Because the return type is always <strong>void</strong>, it isn't mentioned in the syntax. No code is allowed inside the braces ({}).</td>
</tr>
<tr>
<td>eventHandler</td>
<td>myClassInstance.myDelegate += eventHandler(otherClass.myInstanceMethod);</td>
<td>Although the syntax of the <strong>eventHandler</strong> keyword might give the impression that <strong>eventHandler</strong> is an X++ function, it isn't a function. The <strong>eventHandler</strong> keyword tells the compiler that a method is being subscribed to a delegate.</td>
</tr>
</tbody>
</table>
Subscribe or add a method to a delegate

```cpp
myClassInstance.myDelegate += eventHandler(OtherClass::aStaticMethod);
```

In the code, the static method `OtherClass::aStaticMethod` becomes subscribed to the delegate.

Call a delegate

```cpp
myClassInstance.myDelegate("Hello");
```

This call to the delegate prompts the delegate to call each method that is subscribed to the delegate. The subscribed methods are called in the same order in which they were added to the delegate. One subscribed method must be completed before the delegate calls the next method.

**Example**

The two classes in the following code example demonstrate how to define an event, subscribe to an event, and raise an event. The `PointWithEvent` class defines a delegate, `moved`. The `move` method calls the `moved` delegate, thereby notifying any objects that have subscribed to the event. The `PointKeeper` class defines the `writeMove` method and assigns it as the event handler for the `moved` delegate of the `Point` instance created in the `createAndMove` method.
class PointWithEvent
{
    // Instance fields.
    real x;
    real y;

    // Constructor to initialize fields x and y.
    void new(real _x, real _y)
    {
        x = _x;
        y = _y;
    }

    void move(real x_offset, real y_offset)
    {
        x += x_offset;
        y += y_offset;
        this.moved(abs(x_offset) + abs(y_offset));
    }

    delegate void moved(real distance)
    {
    }
}

class PointKeeper
{
    public void createAndMove()
    {
        PointWithEvent point = new PointWithEvent(1.0, 2.0);
        point.moved += eventhandler(this.writeMove);
        point.move(4.0, 5.0);
        // Output is "9.0".
    }

    public void writeMove(real distance)
    {
        info(any2Str(distance));
    }
}

Event handlers and Pre/Post methods

In legacy X++, it was possible to prescribe in metadata that certain methods were to be executed prior to and after the execution of a method. The information about what subscribes call was recorded on the publisher, which isn’t useful in the environment. It’s now possible to provide Pre and Post handlers through code, by providing the SubscribesTo attribute on the subscribers.

Example of pre and post methods
This example shows a publishing method called Publisher. Two subscribers are enlisted with the PreHandlerFor and PostHandlerFor. The code shows how to access the variables, and the return values.

This feature is provided for backward compatibility and, because the application code doesn't have many delegates, to publish important application events. Pre and Post handlers can easily break as the result of added or removed parameters, changed parameter types, or because methods are no longer called, or called under different circumstances. Attributes are also used for binding event handlers to delegates:

In this case, the SubscribesTo attribute specifies that the method RentalFinalizedEventHandler should be called when the FmRentalCheckoutProcessor.RentalTransactionAboutToBeFinalizedEvent delegate is called. Since the binding between the publisher and subscribers is done through attributes, there's no way of specifying the sequence in which subscribers are called.
You can use SQL statements, either interactively or in source code, to retrieve and modify data that is stored in the database. You can use the `select` statement and API methods for these tasks:

- **Select data**: Select the data to view or modify.
  - `select` statement – Fetch records.

- **Insert data**: Add one or more new records to a table.
  - `insert` and `doInsert` methods – Insert one record at a time.
  - `insert_recordset`, `RecordInsertList.insertDatabase`, and `RecordSortedList.insertDatabase` methods – Insert multiple records at the same time.

- **Update data**: Modify the data in existing table records.
  - `update` and `doUpdate` methods – Update one record at a time.
  - `update_recordset` statement – Update multiple records at the same time.

- **Delete data**: Remove existing records from a table.
  - `delete` and `doDelete` methods – Delete one record at a time.
  - `delete_from` statement – Delete multiple records at the same time.

Here are some other statements that you will use in data access:

- while `select` statement
- `select` expression
- `next` statement

**Transactional integrity** helps prevent data corruption and improve scalability.

The [Conversion of operations from set-based to record-by-record topic](#) provides information about how you can use the record set-based statements and methods more efficiently.

You can also use the `SysDa classes` to retrieve and modify data. The extensible SysDa API provides almost all the data access possibilities that are available in X++.

The `executeQueryWithParameters` API can help mitigate a SQL injection attack.

For information about using joins, see [Common misconception about Exists and Notexists joins](#).
The **select** statement fetches or manipulates data from the database.

- All **select** statements use a table variable to fetch records. This variable must be declared before a **select** statement can be run.

- The **select** statement fetches only one record, or field. To fetch or traverse multiple records, you can use the **next** statement or the **while select** statement.
  - The **next** statement fetches the next record in the table. If no **select** statement precedes the **next** statement, an error occurs. If you use a **next** statement, don't use the **firstOnly** find option.
  - It's more appropriate to use a **while select** statement to traverse multiple records.

- The results of a **select** statement are returned in a table buffer variable.

- If you use a field list in the **select** statement, only those fields are available in the table variable.

### Select example

The following example fetches all the columns in the first row of the CustTable table and prints the value in the **AccountNum** column of that row.

```java
CustTable custTable;
select * from custTable;
info("AccountNum: " + custTable.AccountNum);
```

For more examples of data selection, see [Select data](#).

### Insert example

The following example inserts a new record into the CustTable table. The **AccountNum** column of the new record is set to 2000, and the **CustGroup** column is set to 1. Other fields in the record will be blank.

```java
ttsBegin;
   CustTable custTable;
   select forUpdate custTable;
   custTable.AccountNum = '2000';
   custTable.CustGroup = '1';
   custTable.insert();
ttsCommit;
```

For more examples of data insertion, see [Insert data](#).

### Update example

The following example selects the CustTable table for update. Only records where the value of the **AccountNum** field equals 2000 are updated. Because there is no call to **next**, and this example doesn't use a **select while** statement, only one record is updated. The value of the **CreditMax** field is changed to 5000.
ttsBegin;
    CustTable custTable;
    select forUpdate custTable
        where custTable.AccountNum == '2000';
    custTable.CreditMax = 5000;
    custTable.update();
ttsCommit;

For more examples of data updates, see Update data.

Delete example

In the following example, all records in the CustTable table where the AccountNum field equals 2000 are deleted from the database. One record is deleted at a time.

ttsBegin;
    CustTable custTable;
    while select forUpdate CustTable
        where custTable.AccountNum == '2000'
        {
            custTable.delete();
        }
    custTable.delete();
    custTable.update();
ttsCommit;

For more examples of data deletion, see Delete data.

Syntax of the select statement

The following symbols are used in the syntax:

- [] – Brackets enclose an optional element.
- {} – Braces enclose an element that can be included zero or more times.
- + – A plus sign indicates an element that can be included one or more times.
- | – A bar indicates options.

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>EXPRESSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SelectStatement</td>
<td>= select Parameters</td>
</tr>
<tr>
<td>Parameters</td>
<td>= { FindOption } [ FieldList from ] TableBufferVariable \ IndexClause \ Options \ WhereClause \ JoinClause</td>
</tr>
<tr>
<td>FindOption</td>
<td>= crossCompany \ reverse \ firstFast \ FirstOption \ forUpdate \ noFetch \ ForceOption \ forceSelectOrder \ forceNestedLoop \ LockOption \ readableRead \ repeatableRead \ validTimeState</td>
</tr>
<tr>
<td>FirstOption</td>
<td>= firstOnly \ firstOnly10 \ firstOnly100 \ firstOnly1000</td>
</tr>
<tr>
<td>LockOption</td>
<td>= optimisticLock \ pessimisticLock</td>
</tr>
</tbody>
</table>
### SYMBOL

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>EXPRESSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ForceOption</td>
<td>forcePlaceholders</td>
</tr>
<tr>
<td>FieldList</td>
<td>{ Field }</td>
</tr>
<tr>
<td>Field</td>
<td>Aggregate( FieldIdentifier )</td>
</tr>
<tr>
<td>Aggregate</td>
<td>sum</td>
</tr>
<tr>
<td>Options</td>
<td>OrderClause</td>
</tr>
<tr>
<td>OrderClause</td>
<td>[OrderBy [GroupBy]]</td>
</tr>
<tr>
<td>OrderBy</td>
<td>order (by) FieldOrder, FieldOrder</td>
</tr>
<tr>
<td>GroupBy</td>
<td>group (by) FieldOrder, FieldOrder</td>
</tr>
<tr>
<td>FieldOrder</td>
<td>FieldIdentifier [ asc</td>
</tr>
<tr>
<td>IndexClause</td>
<td>index IndexName</td>
</tr>
<tr>
<td>WhereClause</td>
<td>where Expression InClause</td>
</tr>
<tr>
<td>InClause</td>
<td>in List</td>
</tr>
<tr>
<td>JoinClause</td>
<td>[exists</td>
</tr>
<tr>
<td>ContainerVariable</td>
<td>A container.</td>
</tr>
<tr>
<td>Expression</td>
<td>An expression.</td>
</tr>
<tr>
<td>TableBufferVariable</td>
<td>The variable name for the results.</td>
</tr>
<tr>
<td>FieldIdentifier</td>
<td>The name of a field in the table.</td>
</tr>
<tr>
<td>IndexName</td>
<td>The name of an index for a table.</td>
</tr>
<tr>
<td>List</td>
<td>An array of values.</td>
</tr>
</tbody>
</table>

### Aggregate functions

The aggregate functions perform calculations on a single field over a group of records.

- The result is returned in the field that you perform the aggregate function over.
- The fields in the results are the aggregate values and the fields in the `group by` clause.
- You can count, average, or sum only integer and real fields.
- In cases where the `sum` function will return `null`, no rows are returned.
**Differences between X++ and SQL**

In industry-standard SQL, a database query can contain aggregate functions. Examples include `count(RecID)` and `sum(columnA)`. When an aggregate function is used, but no rows match the `where` clause, a row must be returned to hold the result of the aggregates. The row that is returned shows the value 0 (zero) for the `count` function and `null` for the `sum` function. X++ doesn't support the concept of `null` values for the database. Therefore, in cases where the `sum` function will return `null`, no row is returned to the user. Additionally, every data type has a specific value that is treated as a `null` value in some circumstances.

**Grouping and ordering the query results**

A query can have multiple `group by` clauses, but the fields can be qualified by a table name in only one `group by` clause. We recommend that you use table name qualifiers. The `order by` clause follows the same syntax patterns as `group by`. Both clauses, if they are provided, must appear after the `join` (or `from`) clause, and both must appear before any `where` clause that exists on the same `join` clause. We recommend that all `group by`, `order by`, and `where` clauses appear immediately after the last `join` clause. The following example shows a `group by` clause where a field is qualified by a table name.

```csharp
CustTable custTable;
CustGroup custGroup;
struct groupSummary = new struct("int CustomerCount; str CustGroup");

while select count(CreditMax) from custTable
   join custGroup
   order by custGroup.Name
   group by custGroup.CustGroup
   where custTable.CustGroup == custGroup.CustGroup
      && custGroup.Name like "*Days*"
{
   groupSummary.value("CustomerCount", custTable.CreditMax);
   groupSummary.value("CustGroup", custGroup.CustGroup);
   info(groupSummary.toString());
}

// Example output:
// (CustomerCount:1; CustGroup:"1")
// (CustomerCount:3; CustGroup:"2")
```

**Join tables**

The following example shows how an inner join can be performed as part of a `select` statement. The example also shows an `order by` clause where each field is qualified by a table name. Therefore, you can use just one `order by` clause to control how the retrieved records are sorted.
CustTable custTable;
CustGroup custGroup;
struct output = new struct("int AccountNum; str CustGroup");

while select AccountNum from custTable
    join Name from custGroup
    order by custGroup.Name, custTable.AccountNum
    where custTable.CustGroup == custGroup.CustGroup
{
    info(custGroup.Name + ' : ' + custTable.AccountNum);
}

// Example output:
// Days1: 6000
// Days1: 6001
// Days2: 5000

Using where, order by, and index hint together in a query

You use the order by keyword in select statements to order the data that is returned. Use the index hint keyword to specify the index that should be used in the query, and to sort the selected records in the manner that is defined by the index. Indexes optimize the selection of records. To select records in a specific order, combine the index hint keyword with an order by expression. If you want the output to be sorted in reverse order, use the reverse keyword. If a table index has been disabled (that is, if the index’s Enabled property is set to No), the select statement that references the index is still valid. However, the database can't use the index as a hint to sort the data, because the index doesn't exist in the database. The following table shows how to use the index hint and order by keywords in select statements.

<table>
<thead>
<tr>
<th>TASK</th>
<th>SELECT STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select records when the order isn't significant.</td>
<td>select .. where ...</td>
</tr>
<tr>
<td>Select records when the order is significant.</td>
<td>select .. order by ... where ...</td>
</tr>
<tr>
<td>Select records, and force a specific index to be used.</td>
<td>select .. index hint ... where ...</td>
</tr>
<tr>
<td>Select records when the order is significant, and force a specific index to be used.</td>
<td>select .. index hint ... order by ... where ...</td>
</tr>
</tbody>
</table>

The following example shows how to select transactions from the SalesTable table, based on a range of customers and due dates.

SalesTable salesTable;
select salesTable
    index hint CustIdx
    order by CustAccount
where
    salesTable.CustAccount >= '3000'
    && salesTable.CustAccount <= '4000'
    && salesTable.FixedDueDate >= 12\12\2004
    && salesTable.FixedDueDate <= 05\05\2009;

asc keyword

The asc keyword is an option on the order by or group by clause. It specifies an ascending sort order. If neither asc nor desc is specified, the sort is ascending.
avg keyword

The **avg** keyword returns the average of the fields.

```plaintext
CustTable custTable;
select avg(Value) from custTable;
info(int642Str(custTable.Value));
```

count keyword

The **count** keyword returns the number of records.

```plaintext
CustTable custTable;
int64 iCountRows;
select count(RecID) from custTable;
iCountRows = custTable.RecID;
info('Rows: ' + int642Str(iCountRows));
```

crossCompany keyword

The **crossCompany** keyword returns data for all companies that the user is authorized to read from. You can add a container to reduce the number of companies that are involved. The following example returns data for companies that the user is authorized to read from. Results are limited to the dat and dmo companies.

```plaintext
CustTable custTable;
container conCompanies = [ 'dat', 'dmo' ];
select crossCompany: conCompanies * from custTable;
```

crossCompany clause can contain arbitrary expressions

The **crossCompany** clause can be used on **select** statements to indicate the companies that the search statement should take into account. The syntax has been enhanced to allow arbitrary expressions (of type container) instead of a single identifier, which is a variable of type container.

This code examples creates a container with the companies.

```plaintext
private void SampleMethod()
{
   MyTable t;
   container mycompanies = [ 'dat', 'dmo' ];
   select crosscompany: mycompanies t;
}
```

This code uses an expression instead of the variable.
private void SampleMethod()
{
    MyTable t;
    container mycompanies = ['dat', 'dmo'];
    select crosscompany: ([dat] + [dmo]) t;
}

desc keyword

The desc keyword is an option on the order by or group by clause. It specifies a descending sort order. If neither asc nor desc is specified, the sort is ascending.

    CustTable custTable;
    select * from custTable
    order by Value desc;

exists keyword

The exists keyword is a method that returns a Boolean value and a join clause.

    CtrTable ctrTable;
    CustTable custTable;
    while select AccountNum, Value from custTable
        order by AccountNum
        exists join * from ctrTable
        where (ctrTable.AccountNum == custTable_AccountNum)
    {
    

firstFast keyword

The firstFast keyword is a priority hint. The first row appears more quickly, but the total return time for this option might be slower. The firstFast hint is automatically issued from all pages.

The following code example quickly returns the first row.

    CustTable custTable;
    select firstFast custTable
    order by AccountNum;

firstOnly, firstOnly10, firstOnly100, and firstOnly1000 keywords

The firstOnly keywords speed up the fetch by returning a limited number of rows. When you include firstOnly in your query, the runtime returns a table buffer. When you omit firstOnly, the runtime allocates an object that can iterate over records. From a performance perspective, you should use firstOnly only when your intent is to fetch the first record.

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>firstOnly</td>
<td>Return only the first row.</td>
</tr>
<tr>
<td>firstOnly10</td>
<td>Return 10 rows.</td>
</tr>
<tr>
<td>firstOnly100</td>
<td></td>
</tr>
<tr>
<td>firstOnly1000</td>
<td></td>
</tr>
</tbody>
</table>
The following code example returns only the first row of the results.

```csharp
CustTable custTable;
select firstOnly custTable
  index hint AccountIdx
where custTable.AccountNum == '5000';
```

**forceLiterals keyword**

The `forceLiterals` keyword instructs the kernel to reveal the actual values that are used in `where` clauses to the Microsoft SQL Server database at the time of optimization. The `forceLiterals` and `forcePlaceholders` keywords are mutually exclusive. For more information, see the `forcePlaceholders keyword` section.

**WARNING**

You should not use the `forceLiterals` keyword in `select` statements, because it could expose code to an SQL injection security threat.

**forceNestedLoop keyword**

The `forceNestedLoop` keyword forces the SQL Server database to use a nested-loop algorithm to process a particular SQL statement that contains a join algorithm. Therefore, a record from the first table is fetched before any records from the second table are fetched. Typically, other join algorithms, such as hash joins and merge joins, are considered. This keyword is often combined with the `forceSelectOrder` keyword.

```csharp
CustGroup custGroup;
CustTable custTable;

while select forceNestedLoop custGroup
  join custTable
  where custGroup.CustGroup == custTable.CustGroup
{
  Info(custTable.CustGroup);
}
```

**forcePlaceholders keyword**

The `forcePlaceholders` keyword instructs the kernel not to reveal the actual values that are used in `where` clauses to the SQL Server database at the time of optimization. By default, this behavior is used in all statements that aren't `join` statements. The advantage of using this keyword is that the kernel can reuse the access plan for similar statements that have other search values. The disadvantage is that, although the access plan is computed, the fact that data distribution might be uneven isn't considered. The access plan is an on-average access plan. The `forcePlaceholders` and `forceLiterals` keywords are mutually exclusive.

The following example iterates through the `CustGroup` table that is joined with the `CustTable` table.
forceSelectOrder keyword

The **forceSelectOrder** keyword forces the SQL Server database to access the tables in a join in the specified order. If two tables are joined, the first table in the statement is always accessed first. This keyword is often combined with the **forceNestedLoop** keyword.

The following example joins the CustGroup and CustTable tables on the **CustGroup** field.

```sql
CustGroup custGroup;
CustTable custTable;

while select forceSelectOrder custGroup
   join custTable
   where custGroup.CustGroup == custTable.CustGroup
{
   Info(custTable.CustGroup);
}
```

forUpdate keyword

The **forUpdate** keyword selects records for update only. Depending on the underlying database, the records might be locked for other users. The following example selects the **CreditMax** column in the CustTable table for update, for the record where the **AccountNum** value is 2000.

```sql
 ttsBegin;
   CustTable custTable;
   select forUpdate custTable
      where custTable.AccountNum == '2000';
   custTable.CreditMax = 5000;
   custTable.update();
 ttsCommit;
```

group by keyword

The **group by** keyword instructs the database to group selected records by fields.

```sql
CustTable custTable;
while select sum(CreditMax) from custTable
   group by CustGroup
{
   info(custTable.CustGroup + ' ' + int642Str(custTable.CreditMax));
}
```

in keyword

The **in** keyword filters rows where a value is contained in a list.
If you don't use the `in` keyword, the code that you write will resemble the following example.

```java
// This code doesn't use the in keyword.
private CostAmountStdAdjustment calcCostAmountStdAdjustment() {
    InventSettlement inventSettlement;
    select sum(CostAmountAdjustment) from inventSettlement
    where inventSettlement.TransRecId == this.RecId
    && inventSettlement.Cancelled == NoYes::No
    && (inventSettlement.OperationsPosting == LedgerpostingType::purchStdProfit
        || inventSettlement.OperationsPosting == LedgerpostingType::purchStdLoss
        || inventSettlement.OperationsPosting == LedgerpostingType::InventStdProfit
        || inventSettlement.OperationsPosting == LedgerpostingType::InventStdLoss);
    return inventSettlement.CostAmountAdjustment;
}
```

If you use the `in` keyword, the code that you write will resemble the following example.

```java
// This code uses the in keyword.
private CostAmountStdAdjustment calcCostAmountStdAdjustment() {
    InventSettlement inventSettlement;
    container ledgerPostingTypes = this.ledgerPostingTypesForCostAmountStdAdjustmentCalculation();
    select sum(CostAmountAdjustment) from inventSettlement
    where inventSettlement.TransRecId == this.RecId
    && inventSettlement.Cancelled == NoYes::No
    && inventSettlement.OperationsPosting in ledgerPostingTypes;
    return inventSettlement.CostAmountAdjustment;
}
protected container ledgerPostingTypesForCostAmountStdAdjustmentCalculation() {
    return [
        LedgerPostingType::purchStdProfit,
        LedgerPostingType::PurchStdLoss,
        LedgerPostingType::InventStdProfit,
        LedgerPostingType::InventStdLoss
    ];
}
```

**Index Keyword**

The `index` keyword instructs the database to sort the selected records as specified by the index.

```java
CustTable custTable;
while select AccountNum, Value from custTable
    index AccountIdx
{
    Info(custTable.AccountNum + " ": " + int642Str(custTable.Value));
}
```

**Index Hint Keyword**

The `index hint` keyword gives the database a hint to use a specific index to sort the selected records as specified in the index. The database can ignore the hint. An incorrect index hint can greatly affect performance.
Index hints should be applied only to SQL statements that don't have dynamic `where` clauses or `order by` clauses, and where the effect of the hint can be verified.

Before you can use `index hint` in queries, you must call `allowIndexHint(true)` on the table. The default behavior for `index hint` is `false`, and the hint is ignored.

**WARNING**
You should use `index hint` sparingly and with caution, and only when you can be sure that it improves performance. The `index hint` keyword and API let you pass the correct hints when they are required. If you're ever in doubt, avoid using `index hint`.

In the following example, the `AccountIdx` index is used to sort the records in the query on the `CustTable` table.

```csharp
str _accountNum = '111';
CustTable custTable;
custTable.allowIndexHint(true);

while select forUpdate custTable
  index hint AccountIdx
  where custTable.AccountNum == _accountNum
{
}
```

**join keyword**

The `join` keyword is used to join tables on a column that is shared by both tables. The join criteria are specified in a `where` clause, because there is no `on` keyword, such as is found in SQL. This keyword reduces the number of SQL statements that are required if you want to loop through a table and update transactions in a related table. For example, you process 500 records in a table and want to update related records in another table. If you use a nested `while select`, there will be 501 trips to the database. However, if you use a `join`, there will be just one trip to the database.

```csharp
CustTable custTable;
CustGroup custGroup;
int totalCredit;

while select custGroup
  join custTable
  where custGroup.CustGroup == custTable.CustGroup
{
  totalCredit += custTable.CreditMax;
}
```

**maxof keyword**

The `maxof` keyword returns the maximum of the fields.

```csharp
CustTable custTable;
select maxof(CreditMax) from custTable;
info(int642Str(custTable.Value));
```

**minof keyword**
The **minof** keyword returns the minimum of the fields.

```java
CustTable custTable;
select minof(CreditMax) from custTable;
info(int642Str(custTable.Value));
```

### noFetch keyword

The **noFetch** keyword indicates that no records should be fetched now. Typically, this keyword is used when the result of the `select` statement is passed on to another application object, such as a query that performs the actual fetch.

The following example creates a query variable but doesn't fetch the records.

```java
CustTable custTable;
select noFetch custTable
    order by AccountNum;
```

### notExists keyword

The **notExists** keyword is selected only if there are no posts.

```java
CustTable custTable;
CtrTable ctrTable;

while select AccountNum, Value from custTable
    order by AccountNum
    notExists join * from ctrTable
    where (ctrTable.AccountNum == custTable.AccountNum)
{
}
```

### optimisticLock keyword

The **optimisticLock** keyword forces a statement to run by using optimistic concurrency control, even if a different value is set on the table.

```java
CustTable custTable;
select optimisticLock custTable
    where custTable.AccountNum > '1000';
```

### order by keyword

The **order by** keyword instructs the database to sort the selected records by the fields in the `order by` list. The keyword **by** is optional.

```java
CustTable custTable;
select * from custTable
    order by accountNum desc
    where custTable.AccountNum > '100';
info("AccountNum: " + custTable.AccountNum);
```

The following example prints the highest **AccountNum** value in the CustTable table.
outer keyword

The outer keyword returns all rows from the table that is named first, even rows that have no match in the table that is named second. This join is a left outer join. Default values are substituted for any data that could not be obtained from a matching row in the other joined table.

There is no left keyword, and there is no right outer join.

For an inner join, the behavior if you filter on an on clause is the same as the behavior if you filter on the where clause.

The following example is based on two tables. The field types and example data are included. There is a one-to-many relationship between the SalesOrder parent table and the SalesOrderLine child table. For each row in the SalesOrder table, there are 0 (zero) or more rows in the SalesOrderLine table. There are two rows in the SalesOrder table.

<table>
<thead>
<tr>
<th>SALESORDERID (INTEGER, PRIMARY KEY)</th>
<th>DATEADDED (DATE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2010-01-01</td>
</tr>
<tr>
<td>2</td>
<td>2010-02-02</td>
</tr>
</tbody>
</table>

The SalesOrderLine table contains a foreign key field that is named SalesOrderID. This field references the primary key column of the SalesOrder table. A SalesOrderID value of 2 doesn't occur in the data for the SalesOrderLine table.

<table>
<thead>
<tr>
<th>SALESORDERLINEID (STRING, PRIMARY KEY)</th>
<th>QUANTITY (INTEGER)</th>
<th>SALESORDERID (INTEGER, FOREIGN KEY)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>32</td>
<td>1</td>
</tr>
<tr>
<td>BB</td>
<td>67</td>
<td>1</td>
</tr>
<tr>
<td>CC</td>
<td>66</td>
<td>1</td>
</tr>
</tbody>
</table>

The following code has a select statement that reads the two tables. The select statement includes a left outer join clause. Both the join criteria and the data filter are on the where clause. The output from the code is also shown. The second record in the output has a SalesOrderID value of 2. However, that value isn’t present in the SalesOrderLine table. Therefore, some of the fields in the second record have default values: 0 for an integer and
pessimisticLock keyword

The **pessimisticLock** keyword forces a statement to run by using pessimistic concurrency control, even if a different value is set on the table.

```java
CustTable custTable;
select pessimisticLock custTable
where custTable.AccountNum > '1000';
```

repeatableRead keyword

This **repeatableRead** keyword specifies that the current transaction must be completed before other transactions can modify data that has been read by logic inside the current transaction. An explicit transaction is completed at either **ttsAbort** or the outermost **ttsCommit**. For a standalone **select** statement, the transaction duration is the duration of the **select** command. However, the database sometimes enforces the equivalent of **repeatableRead** in individual **select** statements, even if this keyword doesn't appear in your code. (The behavior depends on the method that the database uses to determine whether it should scan the tables.) For more information, see the documentation for the underlying relational database product.

reverse keyword

The **reverse** keyword returns records in reverse order.

```java
CustTable custTable;
select reverse custTable
order by AccountNum;
```
sum keyword

The **sum** keyword returns the sum of the fields. It can be used to sum all accounts, order lines, and so on.

```java
CustTable custTable;
select sum(CreditMax) from custTable;
info(int642Str(custTable.Value));
```

validTimeState keyword

The **validTimeState** keyword selects rows from a table where the **ValidTimeStateFieldType** property is set to a value other than **None**.

```java
CustPackingSlipTransHistory history;
utcDateTime dateFrom, dateTo = DateTimeUtil:utcNow();
anytype recid = -1;
select
    validTimeState(dateFrom, dateTo)
    *
    from history;
recid = history.RecId;
info('RecId:' + int642Str(recid));
```

where keyword

The **where** keyword filters rows from a table where the expression is **true**.

The following example finds a customer that has an **AccountNum** value that is more than 100.

```java
CustTable custTable;
select * from custTable
    where custTable.AccountNum > '100';
info("AccountNum: " + custTable.AccountNum);
```

The following examples prints the lowest **AccountNum** value that is more than 100.

```java
CustTable custTable;
select * from custTable
    order by accountNum
    where custTable.AccountNum > '100';
info("AccountNum: " + custTable.AccountNum);
```

The following example prints the highest **AccountNum** value that is more than 100.

```java
CustTable custTable;
select * from custTable
    order by accountNum desc
    where custTable.accountNum > "100";
info("AccountNum: " + custTable.AccountNum);
```
The `select` statement fetches or manipulates data from the database.

- All `select` statements use a table variable to fetch records. This variable must be declared before a `select` statement can be run.

- The `select` statement fetches only one record, or field. To fetch or traverse multiple records, you can use the `next` statement or the `while select` statement.
  - The `next` statement fetches the next record in the table. If no `select` statement precedes the `next` statement, an error occurs. If you use a `next` statement, don’t use the `firstOnly` find option.
  - It’s more appropriate to use a `while select` statement to traverse multiple records.

- The results of a `select` statement are returned in a table buffer variable.

- If you use a field list in the `select` statement, only those fields are available in the table variable.

The following example fetches all the columns in the first row of the `CustTable` table and prints the value in the `AccountNum` column of that row.

```plaintext
CustTable custTable;
select firstonly custTable; // this is a short notation for 'select firstonly * from custTable;
info("AccountNum: " + custTable.AccountNum);
```

The following example prints the value in the `AccountNum` column of each row in the `CustTable` table.

```plaintext
CustTable custTable;
while select custTable
{ info("AccountNum: " + custTable.AccountNum);
}
```

The following example prints the value in the `AccountNum` column of the first two rows that are returned by the `select` statement.

```plaintext
CustTable custTable;
select custTable;
info("AccountNum: " + custTable.AccountNum);
next custTable;
info("AccountNum: " + custTable.AccountNum);
```

For more examples, see `Select statement`. 
You can use SQL statements, either interactively or in source code, to insert one or more rows into tables that are stored in the database.

- **insert method** – Insert one row at a time.
- **doInsert method** – Insert one row at a time.
- **insert_recordset statement** – Copy multiple records directly from one or more tables into another table in one database trip.
- **RecordInsertList.insertDatabase** – Insert multiple rows at the same time in one database trip. Use this construct when you don't have to sort the data.
- **RecordSortedList.insertDatabase** – Insert multiple rows at the same time in one database trip. Use this construct when you want a subset of data from a specific table, and you want that data to be sorted in an order that doesn’t currently exist as an index.

**RecordSortedList, RecordInsertList, and insert_recordset** let you insert multiple records. By using these methods, you reduce communication between the application and the database. Therefore, you help increase performance. In some situations, record set–based operations can fall back to record-by-record operations. For more information, see Conversion of operations from set-based to record-by-record.

### insert method

The **insert** method inserts one record at a time. It generates values for the **RecId** field and system fields, and then inserts the contents of the buffer (that is, the column values) into the database.

- Don’t use a **select** statement on the table variable before you call the **insert** method.
- The **insert** method doesn’t handle all the key field requirements and table dependencies. You must write code to handle them.

Here is how the **insert** method works:

- Only the specified columns of the rows that have been selected by the query are inserted into the named table.
- The columns of the table that is copied from and the columns of the table that is copied to must be type-compatible.
- If the columns of both tables match in type and order, the column list can be omitted from the **insert** clause.

The following example inserts a new record into the CustGroup table. The **CustGroup** column of the new record is set to **41**. Other fields in the record will be blank.

```java
CustGroup custGroup;
ttsBegin;
custGroup.CustGroup = '41';
custGroup.insert();
ttsCommit;
```

To override the behavior of the **insert** method, use the **doInsert** method.

### doInsert method
The `doInsert` method generates values for the `RecId` field and other system fields, and then inserts the contents of the buffer into the database. Use this method when the `insert` method on the table must be bypassed.

**WARNING**

A call to `doInsert` skips all logic, including database event handlers (for example `oninserting` and `oninserted`), chain-of-command `onInsert()`, and the `insert()` call itself. It's generally considered bad practice to use `doInsert`, and we don't recommend that you use it.

### insert_recordset statement

The `insert_recordset` statement copies data directly from one or more source tables into one destination table in one server trip. It's faster to use `insert_recordset` than an array insert (`RecordInsertList.insertDatabase` or `RecordSortedList.insertDatabase`). However, array inserts are more flexible if you want to handle the data before you insert it. Although `insert_recordset` is a record set-based operator that performs operations on multiple records at a time, it can fall back to record-by-record operations in many situations. For more information, see Conversion of operations from set-based to record-by-record.

In the following syntax for the `insert_recordset` statement, brackets ([ ]) indicate optional elements of the statement.

```plaintext
insert_recordset DestinationTable( ListOfFields )

select ListOfFields1 from SourceTable[ where WhereClause ]

[ join ListOfFields2 from JoinedSourceTable[ where JoinedWhereClause ]]
```

- `ListOfFields` in the destination table must match the list of fields in the source tables. Data is transferred in the order in which it appears in the list of fields. Fields in the destination table that aren't present in the list of fields are assigned 0 (zero) values, as in other areas. System fields, such as `RecId`, are assigned transparently by the kernel in the destination table.
- `WhereClause` and `JoinedWhereClause` are described in the `WhereClause` clause in the `select` statement.

### insert_recordset: Inserting data from another table

In this example, the `Value` column in the `NameValuePair` table is summed for each `Name` value. The results of the aggregation are stored in the `ValueSumByName` table.

```plaintext
ValueSumByName valueSumName;
NameValuePair nameValuePair;

insert_recordset valueSumName (Name, ValueSum)
select Name, sum(Value)
from nameValuePair
group by Name;
```

### insert_recordset: Inserting data from variables

The following example shows that the `insert_recordset` statement can insert variable data.

- Include the `firstonly` keyword to insert only one new record. If you omit `firstonly`, a record is inserted for each record in the `CustTable` table.
- Literals, such as `128` or "this literal string", can't be used in the query as a source of data that is inserted.
- The columns in the source table don't have to correspond to the target table.

In this example, one new record is inserted into the `NameValuePair` table. This record has an `Id` value of 1, a
Name value of Name1, and a Value value of 1.

```java
NameValuePair nameValuePair;
CustTable custTable;

int id_var = 1;
str name_var = 'Name1';
int value_var = 1;

insert_recordset nameValuePair (Id, Name, Value)
select firstonly id_var, name_var, value_var from custTable;
```

**insert_recordset: Inserting data by using a join**

The following example shows a join of three tables on an `insert_recordset` statement that has a subselect. It also shows a `while select` statement that has a similar join. A variable is used to supply the inserted value for one column. The `str` variable must be declared, and it must have a length that is less than or equal to the maximum length of the corresponding database field.

In this example, there is an `insert_recordset` statement for the `tabEmplProj5` table. One of the target fields is named `Description`, and its data comes from the local `sDescriptionVariable` variable. The `insert_recordset` statement succeeds even when the configuration key for the `Description` field is turned off. The system ignores both the `Description` field and the `sDescriptionVariable` variable. Therefore, this code provides an example of configuration key automation. Configuration key automation occurs when the system can automatically adjust the behavior of an `insert_recordset` statement that inserts data into fields that the configuration key is turned off for.
static void InsertJoin42Job(Args _args)
{
    GmTabDepartment tabDept2;
    GmTabEmployee tabEmp1;
    GmTabProject tabProj4;
    GmTabEmployeeProject tabEmpProj5;
    str 64 sDescriptionVariable = "From variable."
    delete_from tabEmpProj5;
    insert_recordset tabEmpProj5
    (  Description
        , EmployeeRecId
        , ProjectRecId
    )
    select sDescriptionVariable, RecId
    from tabEmp1
        join tabDept2
            where tabEmp1 .DepartmentGuid == tabDept2 .DepartmentGuid
        join tabProj4
            where tabDept2 .DepartmentGuid == tabProj4 .DepartmentGuid;
    info(strFmt("%1 == Number of rows inserted.", tabEmpProj5.rowCount()));

    while select tabEmpProj5
        join tabEmp1
            where tabEmpProj5.EmployeeRecId == tabEmp1.RecId
        join tabProj4
            where tabEmpProj5.ProjectRecId == tabProj4.RecId
    {
        info(
            tabEmp1 .EmployeeName
            + " --works on--  "
            + tabProj4 .ProjectName
            + " (" + tabEmpProj5 .Description + ")."
        );
    }
}

/*****************  Actual Infolog output
Message (01:05:41 pm)
4 ==Number of rows inserted.
   Alice --works on--  Project ZZZ (From variable.).
   Alice --works on--  Project YY (From variable.).
   Beth --works on--  Project ZZZ (From variable.).
   Beth --works on--  Project YY (From variable.).
*****************/

Handling DuplicateKeyException exceptions

The following example shows how you can catch a `DuplicateKeyException` exception in the context of an explicit transaction. The exception is thrown when a call to `xRecord.insert` fails because the key value already exists. In the `catch` block, your code can either take corrective action or log the error for later analysis. Your code can then continue without losing all the pending work of the transaction. You can’t catch a `DuplicateKeyException` exception that is caused by a set-based operation such as `insert_recordset`.

This example depends on two tables: `SourceTable` and `DestinationTable`. Each table has one mandatory integer field. The fields are named `SourceKeyField` and `DestinationKeyField`, respectively. A unique index is defined on each key field. The `SourceTable` table must have at least one record in it.

```
static void JobDuplicKeyException44Job(Args _args)
{
    SourceTable   sourceTable; // Must have at least one record.
    DestinationTable destinationTable;
    int countTries = 0;
    int numberAdjust = 0;
```
int newKey;
int inote;
container notes;

// Empty the destination table.
delete_from destinationTable;

// Copy all the records from SourceTable to DestinationTable
insert_recordset destinationTable (destinationKeyField)
    select SourceKeyField from sourceTable order by SourceKeyField asc;

// Copy the records from SourceTable to DestinationTable, one at a time.
// This immediately throws a DuplicateKeyException.
ttsBegin;
try {
    countTries++;
    notes += strFmt("Inside the try block, try count is %1.", countTries);
    while select sourceTable
        order by SourceKeyField asc
    {
        destinationTable.clear();
        newKey = sourceTable.SourceKeyField + numberAdjust;
        destinationTable.DestinationKeyField = newKey;
        notes += strFmt("%1 is the key to be tried.", newKey);
        destinationTable.insert();
        notes += "Success: .insert()";
    }
    ttsCommit;
} catch (Exception::DuplicateKeyException, destinationTable) // Table is optional.
    {
        notes += "Inside the catch block."
        notes += 'Error: ' + infolog.text().strReplace('
', ',');
        if (countTries <= 1)
        {
            notes += "Will issue retry."
            numberAdjust = 1;
            retry; // Erases Infolog.
        } else
        {
            notes += "Aborting the transaction."
            ttsAbort;
        }
    }
for (inote = 1; inote <= conLen(notes); inote++)
    {
        info(conPeek(notes, inote));
    }
/* Output
---- Inside the try block, try count is 1.
---- 11 is the key to be tried.
---- Inside the catch block.
Cannot create a record in DestinationTable (DestinationTable).
The record already exists.
---- Will issue retry.
---- Inside the try block, try count is 2.
---- 12 is the key to be tried.
---- .insert() successful.
*/
You can use SQL statements, either interactively or in source code, to update one or more rows in a table that is stored in the database.

- **update method** – Update the current record with the contents of the buffer. Also update the appropriate system fields.
- **doUpdate method** – Update one row at a time.
- **update_recordset statement** – Update multiple records in one database trip. By using the update_recordset statement, you reduce communication between the application and the database. Therefore, you help increase performance. In some situations, record set-based operations can fall back to record-by-record operations. For more information, see Conversion of operations from set-based to record-by-record.

### update method

The `update` method updates the current record with the contents of the buffer. It also updates the appropriate system fields. The optional `where` clause specifies a condition that the `update` method tests as it processes each row of the table. Only those rows that test `true` against the condition are updated with the new values.

The following example selects the CustTable table for update. Only records where the value of the `AccountNum` field equals 4000 are updated. Because there is no call to `next`, and this example doesn’t use a `select while` statement, only one record is updated. The value of the `CreditMax` field is changed to 5000.

```plaintext
CustTable custTable;
ttsBegin;
    select forUpdate custTable
    where custTable.AccountNum == '4000';
    custTable.CreditMax = 5000;
    custTable.update();
ttsCommit;
```

### doUpdate method

To override the behavior of the `update` method, use the `doUpdate` method. The `doUpdate` method updates the current record with the contents of the buffer. It also updates the appropriate system fields. You should use the `doUpdate` method when the `update` method on the table must be bypassed. The syntax for a `doUpdate` table method is `void doUpdate()`.

### WARNING

A call to `doUpdate` skips all logic, including database event handlers (for example `onUpdating` and `onUpdated`), chain-of-command `onUpdate()`, and the `update()` call itself. It’s generally considered bad practice to use `doUpdate`, and we don’t recommend that you use it.
update_recordset statement

The `update_recordset` operator is a record set–based operator that updates multiple records in one trip to the server. Therefore, the power of Microsoft SQL Server can help improve the performance of some tasks. The `update_recordset` statement resembles `delete_from` in X++ and `update set` in SQL. It doesn't retrieve each record separately by fetching, changing, and updating. Instead, it works on an SQL-style record set on the database server side. If the `update` method is overridden, the implementation falls back to a classic looping construction, where one record at a time is updated. (This behavior resembles the behavior of `delete_from` for deletions.) Therefore, the construction works on temporary tables and whole table–cached tables by using the looping construction.

The following example updates the CustTable table and increments the value in the `CreditMax` column by 1000 for records where the `CreditMax` value is more than 0 (zero).

```
CustTable custTable;
  ttsBegin;
  update_recordset custTable
    setting CreditMax = custTable.CreditMax + 1000
    where custTable.CreditMax > 0;
  ttsCommit;
```

The following example updates multiple columns.

```
CustTable custTable;
  ttsBegin;
  update_recordset custTable
    setting
      CreditMax = custTable.CreditMax + 1000,
      AccountStatement = CustAccountStatement::Always
    where custTable.CreditMax > 0;
  ttsCommit;
```

The following example shows that the `update_recordset` statement supports joins of several tables. Data from the joined tables can be used to assign values to fields in the table that is being updated.

```
TableEmployee tabEmpl;
TableDepartment tabDept;
TableProject tabProj;
update_recordset tabEmpl
  setting
    currentStatusDescription = tabDept.DeptName + ", " + tabProj.ProjName
  join tabDept
    where tabDept.DeptId == tabEmpl.DeptId
  join tabProj
    where tabProj.ProjId == tabEmpl.ProjId;
```
You can use SQL statements, either interactively or in source code, to delete one or more rows from tables that are stored in the database.

- **delete method** – Delete one row at a time.
- **doDelete method** – Delete one row at a time.
- **delete_from statement** – Delete multiple rows at the same time. By using the delete_from statement, you reduce communication between the application and the database. Therefore, you help increase performance. In some situations, this set-based operation can fall back to a record-by-record operation. For more information, see Conversion of operations from set-based to record-by-record.

### delete method

The **delete** method deletes the current record from the database. To use this method, use a **where** clause to specify the rows to delete. One record at a time is then removed from the specified table.

The **delete** method can be overridden. For example, you might want to add extra validation before records are deleted. If you override the **delete** method, you can run the original (base) version of the **delete** method by calling the **doDelete** method. Therefore, a call to the **doDelete** method is equivalent to a call to **super()** in the **delete** method.

In the following example, all records in the NameValuePair table that satisfy the **where** clause (that is, all records where the value of the Name field equals **Name1**) are deleted from the database. One record is deleted at a time.

```java
    ttsBegin;
    NameValuePair nameValuePair;
    while select forUpdate nameValuePair
        where nameValuePair.Name == 'Name1'
    {
        nameValuePair.delete();
    }
    ttsCommit;
```

The following example deletes records from the LedgerJournalTrans table and updates the associated number sequence.
int counter = 0;
str _journalNum = '';
str _voucher = '';
LedgerJournalTrans ledgerJournalTrans;
LedgerJournalTable ledgerJournalTable;

ttsBegin;
while select forUpdate ledgerJournalTrans
    index hint NumVoucherIdx
    where ledgerJournalTrans.journalNum == _journalNum
       && ledgerJournalTrans.voucher == _voucher
{
    ledgerJournalTrans.doDelete();
    counter++;
}
if (counter && ledgerJournalTable.journalType != LedgerJournalType::Periodic)
{
    NumberSeq::release(ledgerJournalTable.voucherSeries, _voucher);
}
ttsCommit;

**doDelete method**

Like the delete table method, the doDelete table method deletes the current record from the database. Use the doDelete method if the delete table method has been overridden, and you want to run the original (base) version of that method instead of the overridden version. Therefore, a call to the doDelete method is equivalent to a call to super() in the delete method.

**WARNING**

A call to doDelete skips all logic, including database event handlers (for example, onDeleting and onDeleted), chain-of-command onDelete(), and the delete() call itself. It's generally considered bad practice to use doDelete, and we don't recommend that you use it.

**delete_from statement**

The delete_from operator is a record set–based operator that removes multiple records at the same time. This approach can be more efficient and faster than an approach that uses the delete method in a loop to delete one record at a time. If you've overridden the delete method, the system interprets the delete_from statement into code that calls the delete method one time for each row that is deleted.

The following example deletes all records in the NameValuePair table where the value in the Name column is Name1.

```cpp
NameValuePair nameValuePair;
delete_from nameValuePair where nameValuePair.Name == 'Name1';
```

In contrast to the previous example, the following example is inefficient, because it issues a separate SQL delete call to the database server for each record. The delete method never deletes more than one record per call.
// Example of inefficient code.
MyWidgetTable tabWidget; // extends xRecord.
ttsBegin;
  while select forUpdate tabWidget
    where tabWidget .quantity <= 100
    { tabWidget.delete(); }
ttsCommit;

**A delete operation that has an inner join**

Inner joins aren't supported on the delete_from statement. Therefore, you can't use the unmodified join keyword on the delete_from statement. However, you can logically perform an inner join by using other techniques.

The following example shows the recommended way to use the delete_from method and inner joins. This example is relatively efficient. It issues a separate delete_from statement for each loop iteration. However, each delete_from statement can delete multiple records (a subset of all the records that the job deletes).

```csharp
MyWidgetTable tabWidget; // extends xRecord.
ttsBegin;
  while select from tabGalaxy
    where tabGalaxy .isTrusted == 0
    { delete_from tabWidget
      where tabWidget .GalaxyRecId == tabGalaxy .RecId; }
ttsCommit;
```

**A delete operation that uses the notexists join keyword**

You can use the notexists join keyword pair in a delete_from statement. The delete_from statements in the following example are efficient. The notexists join clause enables the delete_from statement to delete a specific set of rows. In this example, the delete_from statement removes all parent-order header rows that there are no child-order line rows for. You can also use the exists join clause on the delete_from statement.

```csharp
static void DeleteFromNotexists3bJob(Args _args)
{ 
  GmTabOrderHeader tabOHeader;
  GmTabOrderLine tabOLine;
  AddressState tabAddressState;
  str 127 sOH_Info;
  str 127 sOL_Data;
  int64 i64OHRecId;
  delete_from tabOLine;
  delete_from tabOHeader;
  // Inserts into parent table.
  sOH_Info = "Albert needs tires.";
  insert_recordset tabOHeader (OH_Info)
    select firstOnly sOH_Info from tabAddressState;
  sOH_Info = "Benson wants plastic.";
  insert_recordset tabOHeader (OH_Info)
    select firstOnly sOH_Info from tabAddressState;
  // Obtain a OrderHeader RecId,
  // use it to insert one child row.
  sOL_Data = "4 re-treads.";
  while select firstOnly tabOLine order by OH_Info
    where tabOLine .OH_Info like "A*"
```
i64OHRecId = tabOHeader .RecId;
insert_recordset tabOLine
   (OL_Data ,OrderHeaderRecId)
   select firstOnly
      sOL_Data ,i64OHRecId
   from tabAddressState;
break;
} // Before the delete notexists.
// Display all parent, and then all child rows.
while select tabOHeader
   order by OH_Info
{
   info(strFmt("Before: OHeader:  OH_Info==%1 , RecId==%2"
      ,tabOHeader .OH_Info ,tabOHeader .RecId));
} while select tabOLine
   order by OL_Data
{
   info(strFmt("Before: OLine:  OL_Data==%1 , OrderHeaderRecId==%2"
      ,tabOLine .OL_Data ,tabOLine .OrderHeaderRecId));
} // Delete_From NotExists Join, to remove from the
// parent table all order headers without children.
delete_from tabOHeader
   notexists join tabOLine
   where tabOHeader .RecId ==
      tabOLine .OrderHeaderRecId;
info(strFmt("%1 is the number of childless OHeader records deleted."
   ,tabOHeader.rowCount())); // After the delete notexists.
// Display all parent, and then all child rows.
info("- - - - - - - - - - - - - - -");
while select tabOHeader
   order by OH_Info
{
   info(strFmt("After: OHeader:  OH_Info==%1 , RecId==%2"
      ,tabOHeader .OH_Info ,tabOHeader .RecId));
} while select tabOLine
   order by OL_Data
{
   info(strFmt("After: OLine:  OL_Data==%1 , OrderHeaderRecId==%2"
      ,tabOLine .OL_Data ,tabOLine .OrderHeaderRecId));
} /**************  Actual Infolog output
Message (12:54:14 pm)
Before: OHeader:  OH_Info==Albert needs tires. , RecId==5637144608
Before: OHeader:  OH_Info==Benson wants plastic. , RecId==5637144609
Before: OLine:  OL_Data==4 re-treads. , OrderHeaderRecId==5637144608
  1 is the number of childless OHeader records deleted.
- - - - - - - - - - - - - - -
After: OHeader:  OH_Info==Albert needs tires. , RecId==5637144608
After: OLine:  OL_Data==4 re-treads. , OrderHeaderRecId==5637144608
**************/
A **while select** statement is used to handle data. It’s the most widely used form of the **select** statement. The **while select** statement loops over many records that meet specific criteria, and can run a statement on each record. The syntax of a **while select** statement resembles the syntax of a **select** statement, but the statement is preceded by **while select** instead of **select**.

- Typically, when you use the **while select** statement for data manipulation, you use it in a transaction to ensure data integrity.
- The results of the **while select** statement are returned in a table buffer variable.
- If you use a field list in the **select** statement, only those fields are available in the table variable.
- If you use aggregate functions, such as **sum** or **count**, the results are returned in the fields that you perform the **sum** or **count** over. You can count, average, or sum only integer and real fields.
- The **select** statement itself is run only one time, immediately before the first iteration of the statements in the loop.
- Any Boolean expressions that are added to the **while select** statement (for example, **iCounter < 1**) are tested only one time. This behavior differs from the behavior of the **while** statement in languages such as C++ and C#. For example, the following loop can have more than one iteration.

```java
int iCounter = 0;
BankAccountTable xrecBAT;

while select * from xrecBAT
    where iCounter < 1
{   iCounter++;
    info(strFmt("%1 , %2", AccountID, iCounter, xrecBAT.AccountID));
}
```

The following example prints the **AccountNum** and **SalesGroup** values of every customer in the CustTable table whose account number is within a specified range.

```java
CustTable custTable;

while select custTable
    order by custTable.AccountNum
    where custTable.AccountNum >= '4010' && custTable.AccountNum <= '4100'
{   info(strFmt("%1 , %2", custTable.AccountNum, custTable.SalesGroup));
    custTable = custTable.Next();
}
```
Write select statements as expressions

11/24/2021 • 2 minutes to read • Edit Online

You can use a `select` statement as an expression. This type of `select` statement is known as an `expression select` statement.

- You can't use a table buffer variable in an expression `select` statement.
- The name of the table must be used in the `from` clause.
- The `join` keyword isn't supported.
- The table name can't be used to qualify a field name in the `order by` clause.
- In a `where` clause, the table name must be used as a qualifier of the field.
- You can mention only one table in an expression `select` statement. Therefore, subselects aren't supported as a workaround for the unsupported `join` keyword.
- The only column that can be filled with data is the column that is named before the `from` clause in the `select` clause.
- After the closing parenthesis, the name of a column is used to reference the data value.

The following expression returns the value in the `AccountNum` column of the first row in the `CustTable` table (if a row exists).

```plaintext
str accountNum = (select AccountNum from CustTable order by AccountNum desc).AccountNum;
info('Max AccountNum: ' + accountNum);
```

Here is a simpler way to achieve the same result as the previous example.

```plaintext
str accountNum = (select maxof(AccountNum) from CustTable).AccountNum;
info('Max AccountNum: ' + accountNum);
```

The following example returns the maximum `RecId` value of customers that aren't blocked. Here, the `maxof` aggregate function is used, and the `RecId` field is mentioned in the function. The field that is mentioned in the aggregate function must match the field name that is used to reference the data value after the closing parenthesis. Otherwise, empty data is returned.

```plaintext
int64 nRecId = (select maxof(RecId) from CustTable
    where CustTableBlocked == CustVendorBlocked::No).RecId;
info('Max RecId: ' + int642Str(nRecId));
```

In the following example, the `RecId` field is used to reference a data value that isn't a `RecId` value. The `count` aggregate function doesn't return a `RecId` value. It's a typical practice to use the `RecId` field with the `count` function.

```plaintext
int64 nRecId = (select count(RecId) from CustTable
    where CustTableBlocked == CustVendorBlocked::No).RecId;
info('Count of unblocked customers: ' + int642Str(nRecId));
```

select statements on fields

You can use a `select` statement in a lookup on a field. After a `select` statement that fetches a record in a table,
you can enter `.fieldName` to reference a field in the table. These `select` statements must be used in expressions. A *normal select statement* differs from a *field select statement* in the following way:

- The field `select` statement operates directly on a table.
- The normal `select` statement operates on a table buffer variable.

The following examples shows how to access fields from a select statement.

```java
print((select CustTable order by AccountStatement).AccountStatement);
if (((select custTable where CustTable.AccountNum == '3000').CreditMax < 5000)
{
    info('This customer has a credit maximum less than $5000.');
}
This topic describes transactional integrity in the X++ language.

If you don't take steps to ensure the integrity of transactions, data corruption can occur. At the very least, you might experience poor scalability with respect to concurrent users on the system. Two internal checking features help ensure the integrity of transactions: the forUpdate check and the ttsLevel check.

- A forUpdate check helps ensure that a record can be updated or deleted only if it has first been selected for update. You can select a record for update by using either the forUpdate keyword in the select statement or the selectForUpdate method on the table.
- A ttsLevel check helps ensure that a record can be updated or deleted only in the same transaction scope where it was selected for update.

The following statements are used to help ensure integrity:

- ttsBegin – This statement marks the beginning of a transaction. It helps ensure data integrity and also helps ensure that all updates that are done until the transaction ends (through ttsCommit or ttsAbort) are consistent.
- ttsCommit – This statement marks the successful end of a transaction. It ends and commits a transaction. The Finance and Operations app ensures that a transaction that has been committed will be performed according to intentions.
- ttsAbort – This statement lets you explicitly discard all changes in the current transaction. In this case, the database is rolled back to the original state, where nothing has been changed. Typically, you use this statement if you've detected that the user wants to break the current job. The ttsAbort statement helps ensure that the database is consistent.

Usually, it's a better idea to use exception handling instead of ttsAbort. The throw statement automatically aborts the current transaction. As the following example shows, statements between ttsBegin and ttsCommit can include one or more transaction blocks. In these cases, nothing is committed until a successful exit from the final ttsCommit statement occurs.

```x++
ttsBegin;
   // Some statements.
   ttsBegin;
   // More statements.
   ttsCommit;
ttsCommit;
```

The following example selects a set of records and updates the CustGroup field. This code will throw an exception if the select statement doesn't return any records.

```x++
Custtable custTable;
ttsBegin;
   select forUpdate custTable where custTable.AccountNum == '5000';
   custTable.CustGroup = '1';
   custTable.update();
ttsCommit;
```

Example of code that is rejected by the forUpdate check
In this example, the first failure occurs because the `forUpdate` keyword is missing.

```
ttsBegin;
    select myTable; // Rejected by the forUpdate check.
    mytable.myField = 'xyz';
    myTable.update();
ttsCommit;
```

Example of code that is rejected by the ttsLevel check

In this example, the failure occurs because the transaction scope of the update differs from the transaction scope where the record was selected for update in `ttsCommit`.

```
ttsBegin;
    select forUpdate * from myTable;
    myTable.myField = 'xyz';
ttsCommit;

ttsBegin;
    myTable.update(); // Rejected by the ttsLevel check.
ttsCommit;
```
You can use the following statements and methods to help improve performance by reducing communication between the application and the database:

- delete_from
- update_recordset
- insert_recordset
- RecordSortedList.insertDatabase
- RecordInsertList.insertDatabase

In some situations, these record set–based operations can be converted to slower record-by-record operations. The following table identifies these situations.

<table>
<thead>
<tr>
<th>SITUATION</th>
<th>DELETE_FROM</th>
<th>UPDATE_RECOR_DSET</th>
<th>INSERT_RECORDSET</th>
<th>RECORDSORTED_LIST</th>
<th>RECORDINSERTLIST</th>
<th>USED TO OVERRIDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-SQL tables</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Delete actions</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>skipDeleteActions</td>
</tr>
<tr>
<td>The database log is enabled.</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td>skipDatabaseLog</td>
</tr>
<tr>
<td>Overridden method</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td>skipDataMethods</td>
</tr>
<tr>
<td>Alerts are set up for the table.</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td>skipEvents</td>
</tr>
<tr>
<td>The ValidTimeState FieldType property on a table is set to a value other than None.</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

You can use the skip* settings that are shown in the “Used to override” column to explicitly skip or ignore one or more factors that adversely affect performance. If one of the previously mentioned SQL operations is downgraded to a record-by-record operation, all the skip* settings are ignored. In the following example code, the insert method on the myTable table is run, even though it’s explicitly stated that this method should be skipped if a container or memo field is defined for myTable.
public void tutorialRecordInsertList()
{
    MyTable myTable;
    RecordInsertList insertList = new RecordInsertList(
        myTable.TableId,
        True);
    int i;
    for ( i = 1; i <= 100; i++ )
    {
        myTable.value = i;
        insertList.add(myTable);
    }
    insertList.insertDatabase();
}
This topic explains how to create extensible queries by using the SysDa application programming interface (API).

The extensible SysDa API provides almost all the data access possibilities that are available in X++. In fact, the APIs are wrappers around the code that the X++ compiler would generate. Therefore, use of the SysDa classes carries no overhead, unlike use of the `QueryRun` object, for example. Additionally, the check that the X++ compiler does on data access statements is your responsibility. For example, you create a `where` clause that compares a globally unique identifier (GUID) to an integer. The X++ compiler would diagnose this clause as an error.

The SysDa APIs include an extensive set of APIs for creating custom queries. However, there is a smaller set of types that drives the primary query activities:

- **Select**: `SysDaQueryObject`, `SysDaSearchObject`, and `SysDaSearchStatement`
- **Update**: `SysDaUpdateObject` and `SysDaUpdateStatement`
- **Insert**: `SysDaInsertObject` and `SysDaInsertStatement`
- **Delete**: `SysDaQueryObject`, `SysDaDeleteObject`, and `SysDaDeleteStatement`

The following sections provide examples of each type of query and the customizations that it supports. The examples use a table that is named TestTable. This table has two fields: a string field that is named `stringField` and an integer field that is named `intField`.

**Select query**

To run a `select` query, follow these steps.

1. Create and configure a `SysDaQueryObject` object that specifies the table instance that will contain the designated records.
2. Create a `SysDaSearchObject` object, and pass the `SysDaQueryObject` object to the constructor.
3. Iterate over the results of the query by passing the `SysDaSearchObject` object to the `SysDaSearchStatement.findNext()` method.

The following example finds all rows in TestTable where `intField` <= 5.
// t is the table buffer that will hold the result.
TestTable t;

// Create the query.
var qe = new SysDaQueryObject(t);

// Add clauses to the query. First the projection.
var s = qe.projection()
    .add(fieldStr(TestTable, intField))
    .add(fieldStr(TestTable, stringField));

// At this point the query is:
// intField, stringField FROM TestTable

// Add a where clause to include rows where intField is <= 5.
qe.WhereClause(new SysDaLessThanOrEqualsExpression(
    new SysDaFieldExpression(t, fieldStr(TestTable, intField)),
    new SysDaValueExpression(5)));

// Now the query is:
// intField, stringField FROM TestTable WHERE (TestTable.intField<= 5)

// Order the results by intField.
qe.OrderByClause().addDescending(fieldStr(TestTable, intField));

// Now the query is:
// intField, stringField FROM TestTable ORDER BY intField DESC WHERE (TestTable.intField<= 5)

var so = new SysDaSearchObject(qe);
var ss = new SysDaSearchStatement();

// Enumerate the designated values by using ss.
while (ss.findNext(so))
{
    info(t.stringField);
}

---

**Update statement**

To run an update statement, follow these steps.

1. Create and configure a `SysDaUpdateObject` object.
2. Update data by passing the `SysDaUpdateObject` object to the `SysDaUpdateStatement.execute()` object.
   Because updates modify the data in the database, you must wrap the call to `execute` in `ttsbegin` and `ttscommit` statements.

The following example updates `stringField` to “fifty” for all rows where `intField = 50`. 
TestTable t;

// Create an update query to find rows where intField = 50.
var uo = new SysDaUpdateObject(t);

// Set stringField to "fifty".
uo.settingClause()
  .add(fieldStr(TestTable, stringField), new SysDaValueExpression("fifty"));

// At this point the update statement is:
// UPDATE_RECORDSET TestTable SETTING stringField=fifty
uo.whereClause(new SysDaEqualsExpression(
    new SysDaFieldExpression(t, fieldStr(TestTable, intField)),
    new SysDaValueExpression(50)));

// Now the update statement is:
// UPDATE_RECORDSET TestTable SETTING stringField=fifty WHERE (TestTable.intField == 50)

// Update the rows.
ttsbegin;
    new SysDaUpdateStatement().execute(uo);
ttscommit;

// Verify the results of the update query.
TestTable t1;
select intField, stringField from t1 where t1.intField == 50;
info("Updated value is: " + t1.stringField);
// Output is: "Updated value is: fifty".

---

**Insert statement**

To run an insert statement, follow these steps.

1. Create and configure a `SysDaInsertObject` object to specify which fields are updated during the insertion.
2. Create and configure a `SysDaQueryObject` object that specifies the source of the rows to insert. The order of the fields in `SysDaQueryObject.projection()` must match the order of the fields in `SysDaInsertObject.fields()`.
3. Assign the `SysDaQueryObject` object to the `SysDaInsertObject` object.
4. Insert the new row by passing the `SysDaInsertObject` object to the `SysDaInsertStatement.executeQuery()` method.

The following example inserts rows where `intField = 40` and `stringField = "en-us"` into TestTable.
TestTable t;

// Specify the fields in the new row.
var insertObject = new SysDaInsertObject(t);
insertObject.fields()
    .add(fieldStr(TestTable, stringField))
    .add(fieldStr(TestTable, intField));

// At this point the insert statement is:
// INSERT_RECORDSET TestTable(stringField, intField) SELECT

// Retrieve the data to insert from the LanguageTable by using a query.
LanguageTable source;
var qe = new SysDaQueryObject(source);
var s1 = qe.projection()
    .Add(fieldStr(LanguageTable, LanguageId))
    .AddValue(40);

// The query statement is:
// LanguageId, 40 FROM LanguageTable

qe.WhereClause(new SysDaEqualsExpression(
    new SysDaFieldExpression(source, fieldStr(LanguageTable, LanguageId)),
    new SysDaValueExpression("en-us")));

// Now the query is:
// LanguageId, 40 FROM LanguageTable WHERE (LanguageTable.LanguageId == en-us)

// Assign the query to the insert statement.
insertObject.query(qe);

// The insert statement is now:
// INSERT_RECORDSET TestTable(stringField, intField) SELECT LanguageId, 40 FROM LanguageTable WHERE (LanguageTable.LanguageId == en-us)

var insertStmt = new SysDaInsertStatement();
ttsbegin;
    insertStmt.executeQuery(insertObject);
ttscommit;

// Verify the results of the insert query.
TestTable t1;
select * from t1 where t1.stringField == "en-us";
info(any2Str(t1.intField) + ":" + t1.stringField);
// The output is "40:en-us".

Delete statement

To run a delete statement, follow these steps.

1. Create and configure a SysDaQueryObject object to specify which rows to delete.
2. Create a SysDaDeleteObject object, and pass the SysDaQueryObject object to the constructor.
3. Delete the rows by passing the SysDaDeleteObject object to the SysDaDeleteStatement.executeQuery() method.

The following example deletes rows where intField is an even number.
TestTable t;

// Build the query that specifies which rows to delete.
var qe = new SysDaQueryObject(t);

var s = qe.projection()
  .add(fieldStr(TestTable, intField));

// At this point the query is:
// intField FROM TestTable

// Delete rows where intField is even.
qe.WhereClause(new SysDaEqualsExpression(
    new SysDaModExpression(
        new SysDaFieldExpression(t, fieldStr(TestTable, intField)),
        new SysDaValueExpression(2)),
    new SysDaValueExpression(0));

// Now the query is:
// intField FROM TestTable WHERE ((TestTable.intField MOD 2) == 0)
var ds = new SysDaDeleteStatement();
var delobj = new SysDaDeleteObject(qe);

// The deletion statement, from the SysDaDeleteObject, is:
// DELETE FROM intField FROM TestTable WHERE ((TestTable.intField MOD 2) == 0)
ttsbegin;
  ds.executeQuery(delobj);
ttscommit;
info("Number of rows after deletion: " + any2Str(t.RowCount()));

Clauses
SysDa queries support several clauses:

- **whereClause** – The **where** clause is constructed from objects that inherit from SysDaQueryExpression. Examples are SysDaEqualsExpression, SysDaNotEqualsExpression, and SysDaLessThanExpression. You can find the full list by filtering in Application Explorer.
- orderByClause
- groupByClause
- joinClause with joinClauseKind
- joinedQuery
- settingClause

Troubleshooting
You can use the **toString()** method on SysDaQueryObject, SysDaUpdateObject, SysDaInsertObject, and SysDaQueryObject objects to view the statement that you’re building.
An SQL injection attack occurs when malicious data values are passed to Microsoft SQL Server in a query string. Those values can cause lots of damage in a database. SQL injection can occur if you aren't careful about how you use a query to pass data that comes from an uncontrolled source, such as user input, to SQL Server. SQL injection isn't usually an issue in Finance and Operations apps, because the built-in data access statements in X++ prevent it. However, if you use Direct-SQL, SQL injection can occur when raw SQL code is passed to the server.

A new API will help mitigate these attacks. The API is available starting with platform updates for version 10.0.17 of Finance and Operations apps (April 2021).

The issue

Consider a scenario where a developer writes the following code to look up the first name of customers, based on their last name.

```csharp
public str GetFirstName(str name)
{
    str sqlStatementText = "SELECT TOP(1) firstName FROM Customer WHERE customer.Name = '" + name + '";

    // Create a connection to the SQL Server
    var connection = new Connection();

    // Create a statement and submit the sql statement to the server:
    Statement statement = connection.createStatement();
    var results= statement.executeQuery(sqlStatementText);

    // Get the first record:
    results.next();
    statement.close();

    // Harvest the results.
    return results.getString(1);
}
```

Additionally, there is either a page where users can enter customer names in a string field, or a service endpoint that enables names to come into the server.

In this scenario, everything works well if users enter valid names such as "Jones." However, a malicious user might enter the following string as a name.

```
'; drop table Customer --
```

In this case, here is the final query that the server runs.
The first quotation mark in the given string just ends the string literal that should contain the name that the user is looking for. Then another SQL statement is run because of the semicolon (;), which is a statement terminator token. This second statement irretrievably deletes the Customer table and all the data in it. Finally, the commenting characters (--) ensure that the single quotation mark at the end doesn't cause syntax errors. Therefore, the string is valid Transact SQL (T-SQL).

SQL injection occurs because the connection to SQL Server doesn't impose any restrictions that prevent it from performing operations that create or delete tables, views, and stored procedures at runtime. Therefore, organizations must rely on the assumption that developers are reasonable people who know what they are doing.

The solution

SQL Server mitigates the threat by using statement parameters. Statement parameters never use literals that are subject to textual changes to the resulting string. Instead, they provide named parameters, the actual content of which is provided contextually. For this release, Microsoft has added a new API that lets you use parameters instead of building SQL strings in code.

The following example shows what the code from the previous example looks like after these changes are incorporated.

```csharp
public str GetFirstName(str name)
{
    str sqlStatementText = "SELECT TOP(1) firstName FROM Customer WHERE customer.Name = @Name";

    // Create a connection to the SQL Server
    var connection = new Connection();

    // Submit the sql statement to the server:
    Statement statement = connection.createStatement();

    // Create a mapping from parameter names onto values
    Map paramMap = SqlParams::create();
    paramMap.add('Name', name);

    // Execute the query, providing both the query
    // and the parameters.
    var results = statement.executeQueryWithParameters(sqlStatementText, paramMap);

    // Capture the results:
    results.next();
    statement.close();

    return results.getString(1);
}
```

The updated example uses the new `executeQueryWithParameters` API instead of the old API that didn't take parameters. The code builds the map that contains the mapping from parameter names to parameter values. In this case, `Name` will be the value of `@Name` in SQL. The incoming `name` value can be anything.

A related method on the `Statement` type is used to run statements that return integer values instead of rows. Typically, the integer value indicates the number of rows that are affected. The following example uses the X++ data statements with the `executeQueryWithParameters` API.
public void InsertWithStrParameter()
{
    var connection = new Connection();
    Statement statement = connection.createStatement();

    connection.ttsbegin();

    str sql = @"UPDATE Wages
    SET Wages.Wage = Wages.Wage * @percent
    WHERE Wages.Level = @Level";

    Map paramMap = SqlParams::create();
    paramMap.add('percent', 1.1);        // 10 percent increase
    paramMap.add('Level', 'Manager');    // Management increase

    int cnt = statement.executeUpdateWithParameters(sql, paramMap);
    statement.close();

    connection.ttscommit();
}

Conclusion

As Microsoft introduces the new methods, we are also marking the existing methods (that is, the methods without the parameters) as obsolete. The usual deprecation periods apply. Therefore, you can update your code to take advantage of the new protection that the parameters provide.

Although the new executeQueryWithParameters API helps you protect your customers from disasters, you aren’t required to use it. You can still do string concatenations and provide an empty parameter set. However, in this case, you don’t gain the advantages that the parameters provide. We hope that you will take this opportunity to eliminate any dangerous usage that you have in your code.
This topic describes how to create and use macros in X++.

Precompiler directives are processed before the code is compiled. The directives declare and handle macros and their values. The directives are removed by the precompiler so that the X++ compiler never encounters them. The X++ compiler only sees the sequence of characters written into the X++ code by the directives.

**WARNING**

Use of macros is not recommended. Macros are supported for backwards compatibility only.

Instead of macros, use language constructs like these:

- Constants
- Sysda for queries.

### #define and #if directives

All precompiler directives and symbols begin with the `#` character. A macro can be defined at any point in the code. The variable can have a value that is a sequence of characters, but it is not required to have a value. The `#define` directive tells the precompiler to create the macro variable, including an optional value. The `#if` directive tests whether the variable is defined, and optionally, whether it has a specific value. The X++ precompiler directives, the macro names that they define, and the `#if` directive value tests are all case-insensitive. However, it is a best practice to begin macro names with an uppercase letter.

### #undef directive

You can use the `#undef` directive to remove a macro definition that exists from a previous `#define`. After a macro name has been created by `#define` and then removed by `#undef`, the macro can be created again by another `#define`. `#undef` has no effect on macros that are created by the `#localmacro` directive.

### Use a Macro Value

You can define a macro name to have a value. A macro value is a sequence of characters. A macro value is not a string (or `str`) in the formal sense of a data type. You assign a value to a macro by appending the value enclosed in parentheses at the end of a `#define` directive. You can use the macro symbol where you want the value to occur in the X++ code. A macro symbol is the name of the macro with the `#` character added as a prefix. The following code sample shows a macro symbol `#MyMacro`. The symbol is replaced by the value of the macro.

### Test a Macro Value

You can test a macro to see whether it has a value. You can also test to see whether its value is equal to a specific sequence of characters. These tests enable you to conditionally include lines of code in your X++ program. There is no way you can test whether a defined macro has a value. You can only test whether a specific value matches the value of a macro. As a best practice, any macro name that you define should always have a value, or it should never have a value. When you alternate between these modes, your code becomes difficult to understand. For macros that have a value, you can vary the value when you see fit.
#defInc and #defDec directives

`#defInc` and `#defDec` are the only directives that interpret the value of a macro and they apply only to macros that have a value that can be converted to the formal `int` type. The value can only contain numerals. The only non-numeric character allowed is a leading negative sign (`-`). The integer value is treated as an X++ `int`, not as an `int64`. For macro names that are used by the `#defInc` directive, it is important that the `#define` directive that creates the macro not reside in a class declaration. The behavior of `#defInc` in these cases is unpredictable. Instead, such macros should be defined in only a method. We recommend that the `#defInc` and `#defDec` directives only be used for macros that have an integer value. The precompiler follows special rules for `#defInc` when the macro value is not an integer, or when the value is unusual or extreme. The following table lists the values that `#defInc` converts to zero (0) and then increments. When a value is converted to 0 by `#defInc`, the original value cannot be recovered, not even by `#defDec`.

<table>
<thead>
<tr>
<th>MACRO VALUE</th>
<th>BEHAVIOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>(+55)</td>
<td>The positive sign (+) prefix makes the precompiler treat this as a non-numeric string. The precompiler treats all non-numeric strings as 0 when it handles a <code>#defInc</code> (or <code>#defDec</code>) directive.</td>
</tr>
<tr>
<td>(&quot;3&quot;)</td>
<td>Integers enclosed in quotation marks are treated as 0. The quotation marks are discarded, and these changes persist.</td>
</tr>
<tr>
<td>()</td>
<td>A string of spaces is treated as 0, and then incremented.</td>
</tr>
<tr>
<td>()</td>
<td>A zero-length string is treated as 0, and then incremented, when the value is enclosed in parentheses, as in <code>#define.MyMac()</code>.</td>
</tr>
<tr>
<td>(Random string.)</td>
<td>Any non-numeric string of characters is treated as 0, and then incremented.</td>
</tr>
<tr>
<td>(0x12)</td>
<td>Hexadecimal numbers are treated as non-numeric strings. Therefore they are converted to 0, and then incremented.</td>
</tr>
<tr>
<td>(-44)</td>
<td>Negative numbers are acceptable, including integers without the negative sign ( - ).</td>
</tr>
<tr>
<td>(2147483647)</td>
<td>The maximum positive <code>int</code> value is changed to the minimum negative <code>int</code> value by <code>#defInc</code>.</td>
</tr>
<tr>
<td>(999888777666555)</td>
<td>Any large number, beyond the capacity of <code>int</code> and <code>int64</code>. This is treated as the maximum positive <code>int</code> value.</td>
</tr>
<tr>
<td>(5.8)</td>
<td>Real numbers are truncated by <code>#defDec</code> (and <code>#defInc</code>). Subsequent symbol substitution shows that the truncation persists.</td>
</tr>
<tr>
<td></td>
<td>When no value and no parentheses are provided for the directive <code>#define.MyValuelessMacro</code>, the precompiler rejects use of the directive <code>#defInc :MyValuelessMacro</code>.</td>
</tr>
</tbody>
</table>

#globaldefine directive
The #globaldefine directive is similar to the #define directive. The difference is that #define directives generally take precedence over #globalmacro directives. This is true regardless of which directive occurs first in the X++ code. A #globaldefine never overwrites a #define directive that has both a macro name and a value. A #globaldefine can overwrite another #globaldefine. A #define directive that has only a name does not overwrite a #globalmacro that has both a name and a value. It is recommended that you use #define, and that you do not use #globaldefine. Use of #globaldefine can create uncertainty that makes code difficult to maintain. The exact semantics of #globaldefine cannot be achieved through #if test directives. By using #if tests you can avoid overwriting a #define and a #globaldefine. But #if tests cannot distinguish between #define and #globaldefine macros.

### Macro parameters

You can define macro values to include parameter symbols. The first parameter symbol is %1, the second is %2, and so on. You pass values for the parameters when you reference the macro symbol name for expansion. Macro parameter values are character sequences of no formal type, and they are comma delimited. There is no way to pass in a comma as part of a parameter value. The number of parameters passed can be less than, greater than, or equal to the number of parameters that the macro value is designed to receive. The system tolerates mismatches in the number of parameters passed. If fewer parameters are passed than the macro expects, each omitted parameter is treated as a zero-length sequence of characters.

#### #localmacro and #globalmacro directives

The #localmacro directive is a good choice when you want a macro to have a value that is several lines long, or when your macro value contains a closing parenthesis. The #localmacro directive is a good choice when you want your macro value to be lines of X++ or SQL code. The #localmacro directive can be written as #macro. However, #localmacro is the recommended term. Both macros have the same behavior. By using the #if directive, you can test whether a macro name is declared with the #define directive. However, you cannot test whether the macro name is declared with the #localmacro directive. Only macros declared by using the #define directive are affected by the #undef directive. In a #define directive, you can specify a name that is already in scope as a #localmacro. The effect is to discard the #localmacro and create a #define macro. This also applies to the opposite sequence, which means that a #localmacro can redefine a #define. A #localmacro (that has both a macro name and a value) always overrides a previous #localmacro that has the same name. However, you cannot always be sure whether the override occurs when you use #globalmacro. For this reason we recommend that you do not use #globalmacro. This same problem occurs with #globaldefine. The main difference between a #define macro and a #localmacro macro is in how their syntax is terminated. The terminators are as follows:

- #define – is terminated by – }
- #localmacro – is terminated by – #endmacro

#localmacro is a better choice for macros with multiple line values. Multiple line values are typically lines of X++ or SQL code. X++ and SQL contain lots of parentheses, and these would prematurely terminate a #define. Both #define and #localmacro can be declared and terminated on either a single line or on subsequent lines. In practice, the #define is terminated on the same line that it is declared on. In practice, the #localmacro is terminated on a subsequent line. Where both macro names and values are supplied, the #globalmacro directive cannot override the #define directive. Also, the #globaldefine directive cannot override the #localmacro directive.

### Nesting Macro Symbols

You can nest precompiler definition directives inside an outer definition directive. The main definition directives are #define and #localmacro.
A `#define` directive can be given inside a `#localmacro` directive, and a `#localmacro` can be inside a `#define`.

### `#macrolib` directive

In the Application Explorer under the Macros node, there are many library nodes that contain sets of macro directives. Both `#define` and `#localmacro` often appear in the contents of these macro libraries. You can use the `#macrolib.MyAOTMacroLibrary` to include the contents of a macro library in your X++ code. The `#if` and `#undef` directives do not apply to `#macrolib` names. However, they do apply to `#define` directives that are the contents of a `#macrolib` macro. The directive `#macrolib.MyAOTMacroLibrary` can also be written as `#MyAOTMacroLibrary`. The `#macrolib` prefix is recommended because it is never ambiguous to a person who later reads the code.

### `#linenumber` Directive

You can use the `#linenumber` directive during your development and debugging of code. It is replaced by the physical line number in the code file.

### Range (scope) of macros

The range in which a macro can be referenced depends on where the macro is defined. In a class, macros that are defined in the parent class can be referenced, but macros defined in a child class cannot be referenced. When the precompiler handles a child class, the precompiler first traces the inheritance chain to the most ascendant class. The precompiler processes all the directives from the class declaration part of the ascendant class. It stores all the macros and their values in its internal tables. The precompiler handles the next class in the inheritance chain the same way. The results of the directives in each class declaration are applied to the internal tables that are already populated from directives that were found earlier in the inheritance chain. When the precompiler reaches the target child class, it again handles the class declaration part. However, it next handles each method in a series of separate operations. The precompiler updates its internal tables in a way that the state of the tables can be restored as they were before processing of the current method began. After the first method is handled, the internal tables are restored before the next method is handled.

### The Method is All Contents of the Node

In this context, a method is defined as the contents of a method node in the Application Object Tree (AOT). In the AOT, you can expand the Classes node, expand a class node, right-click a method node, and then select Edit. Then you can add a line for `#define.MyMacro("abc")` before the method declaration. The precompiler treats this `#define` directive as part of the method, even though the `#define` occurs outside the `{}` block of the method.
This topic describes the use of attributes in X++.

An attribute is a non-abstract class that extends (inherits from) the **SysAttribute** class. Attributes represent or store metadata about types and methods. An attribute can be attached to a class, a class field, a class method, an interface, or a table.

Attributes are applied to the handlers of delegates and methods, to map the handlers to those targets.

### Creating an attribute class

An attribute class can extend the **SysAttribute** class directly, or it can extend any descendant of the **SysAttribute** class. The **SysAttribute** class cannot be used as an attribute because it is declared **abstract**. The following example shows the declaration and design of an ordinary attribute class that you could create.

```java
public class PracticeAttribute extends SysAttribute {
    // Fields in the classDeclaration.
    StartEnd startEndEnum;
    str reason;
    // Constructor.
    public void new(StartEnd _startEndEnum, str _reason) {
        startEndEnum = _startEndEnum;
        reason = _reason;
    }
    // Other methods can go here.
}
```

### Decorating a class with an attribute

The following example shows a class and a method that are decorated with the **PracticeAttribute** given in the previous example. If the constructor of the attribute takes no parameters, the parentheses for the parameters are optional. The attribute decoration could be `[AnotherAttribute]` without parentheses.

```java
[PracticeAttribute(StartEnd::End, "Use the RegularClass class at the end.")]
public class RegularClass {
    [PracticeAttribute(StartEnd::Start, "Use the rehearse method at the start.")]
    public int rehearse() {
        // Logic goes here.
    }
    // More fields and methods belong here.
}
```

You can omit the suffix of the attribute name if the suffix is **Attribute**. For example, you could use `[Practice]` instead `[PracticeAttribute]` in the preceding example.

### Attribute constructors

You can enable your attribute class to store tailored metadata each time it is used to decorate a class, by having its constructor take parameters. The parameters for the constructor must be literals of the primitive types, such as **int**, **enum**, or **str**. The compiler does not construct an instance of the attribute class. It stores the name of the
attribute class, plus the literal values for its constructor. Therefore, if the logic in an attribute constructor would
throw an exception, the exception would not be found by decorating a class with the attribute. The exception
would be found later when a process looks at a class to see the attribute it is decorated with. That is when the
attribute is constructed.

Naming conventions
All attribute classes have the suffix Attribute in their name. The Attribute suffix is the name convention that we
recommend, but it is not a system requirement. You can determine whether a class extends directly from
SysAttribute by selecting the class in the Application Explorer and reviewing the Extends property in the
Properties window.

SysObsoleteAttribute
The system provides several attributes, including the SysObsoleteAttribute class. One use of the
SysObsoleteAttribute class is to notify the compiler that the compile should fail if a particular method is
called in the source code. The compiler rejects the compile, and displays the specific message that is stored in
this use of the attribute. The SysObsoleteAttribute class can also be used to notify the compiler to issue
warning messages instead of errors.

SysObsoleteAttribute code example

```csharp
[SysObsoleteAttribute("The Automobile class might have faster performance.", false)]
class Bicycle
{
    // Members of the Bicycle class go here.
}
```

Metadata reflection
You use reflection to find the attribute metadata that is attached to a class. The classes to use for attribute
reflection are as follows:

- DictClass class – For classes and interfaces.
- DictMethod class – For methods on classes, interfaces, or tables.

On the previous reflection classes, the methods for reflecting on attribute metadata are as follows:

- getAllAttributes method
- getAttribute method
- getAttributedClasses method
- getAttributes method

**NOTE**
There is no mechanism for listing all methods or classes that are adorned with a particular attribute from X++ code.
However, because the X++ compiler records this information in the cross reference database, the information can be
mined from there.

Metadata reflection code example
You use the DictMethod class to find the metadata value of an attribute that is decoration on a method. The
following code example uses the SysEntryPointAttribute class as the attribute. It accepts your parameter
values for the method name, and for the name of the class that contains the method. The parmChecked
method is particular to the SysEntryPointAttribute class, and it is not inherited from its base class
SysAttribute. Each attribute class can have its own method name for its metadata.
static public int MetadataOfSysEntryPointAttributeOnMethod
(
    str _sNameOfClass,
    str _sNameOfMethod
)
{
    // Return Values:
    // 0 == Has the attribute, its metadata value is false;
    // 1 == Has the attribute, its metadata value is true;
    // 2 == The method lacks the SysEntryPointAttribute.
    int nReturnValue = -1,
    nClassId;
    boolean boolParmChecked;
    DictMethod dm;
    Object attributeAsObject;
    SysEntryPointAttribute sepAttribute;
    Global::info("Starting AttributeReflection" + " ::MetadataOfSysEntryPointAttributeOnMethod ....");
    Global::info(strFmt("Parameters are: _sNameOfClass = %1 ,  _sNameOfMethod = %2 .", _sNameOfClass, _sNameOfMethod));
    nClassId = Global::className2Id(_sNameOfClass);
    dm = new DictMethod
        (UtilElementType::ClassInstanceMethod,
        nClassId,
        _sNameOfMethod
    );
    attributeAsObject = dm.getAttribute("SysEntryPointAttribute");
    if (attributeAsObject is SysEntryPointAttribute)
    {
        sepAttribute = attributeAsObject as SysEntryPointAttribute;
        boolParmChecked = sepAttribute.parmChecked();
        if (boolParmChecked)
            nReturnValue = 1;
        else
            nReturnValue = 0;
        Global::info(
            strFmt("Return value is %1.", nReturnValue)
        );
    }
    else
    {
        nReturnValue = 2;
        Global::error("Object is not a SysEntryPointAttribute??");
    }
    return nReturnValue;
}

/***
 * Output displayed in the Infolog.
 Message (05:03:22 pm)
 Starting AttributeReflection ::MetadataOfSysEntryPointAttributeOnMethod ....
 Parameters are: _sNameOfClass = CustCustomerService ,  _sNameOfMethod = create .
 Return value is 1.
 ***/

/**************
// Simple AOT > Jobs job to run the method.
static void AttributeReflection33Job(Args _args)
{
    AttributeReflection::MetadataOfSysEntryPointAttributeOnMethod
        ("CustCustomerService", "create");
}
**************/

This topic compares X++ and C# syntax and programming.

### X++, C# Comparison: Hello World

This section compares the simplest X++ program to its counterpart in C#.

#### X++ to C# Comparisons

The following sections describe some basic similarities and differences between X++ and C#.

### Similarities

The following X++ features are the same for C#:

- Single line (`//`) and multi-line (`/* */`) comments.
- `==` (equal) operator for determining whether two values are equal.
- `!=` (not equal to) operator for determining whether two values are not equivalent.
- `+` (plus sign) operator for string concatenation.

### Differences

The following table lists X++ features that are different in C#.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>X++</th>
<th>C#</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>if</code> and <code>else</code> conditional statements</td>
<td>The <code>if</code> statement accepts any type of expression that it can automatically convert to a Boolean. Common examples include an `int for which 0 means false, or an object for which null means false.</td>
<td>The <code>if</code> statement requires a Boolean expression.</td>
<td>The syntax structure regarding curly braces and parentheses is exactly the same between X++ and C#.</td>
</tr>
</tbody>
</table>
| Literal string | A literal string can be delimited by either of the following:  
- A pair of double quotation mark (") characters.  
- A pair of single quotation mark ('') characters. | A literal string must be delimited by a pair of double quotation mark (") characters. | For X++, the double quotation mark characters are usually used to delimit strings. However, it is convenient delimit a string with single quotation mark characters when your string must contain a double quotation mark character. |
| `char` type | There is no `char` or character type in X++. You can declare a `str` of length one, but it is still a string:
```csharp
str 1 myString = "a";
```
| There is a `char` in C#. You cannot pass a `char` as the parameter to a method that inputs a `string` parameter, although you can first explicitly convert the `char` to a `string`. | For more information about X++ data types, see Primitive Data Types. |
Output of messages

X++ delivers messages to the user in the Infolog window. Common methods include the following:
- The `print` statement:
- static methods on the `Global` class:
  - `Global::info`
  - `Global::warning`
  - `Global::error`

For a command line program, messages can be delivered to the console. Common methods include the following:
- `Console.Out.WriteLine`
- `Console.Error.WriteLine`

---

### X++ and C++ Samples

This section contains two simple code samples. One sample is written in X++, and the other is in C#. Both samples achieve the same result. The following X++ features are demonstrated:

- `//` single line comment
- `/* */` multi-line comment
- `if` statement
- `==` operator
- `!=` operator
- `+` operator to concatenate strings
- `Global::info` for message output, with and without the `Global:` prefix
- `Global::error` for message output
- The use of single and double quotation characters (`'` and `"`) as string delimiters.

---

**NOTE**

The best practice is to use double quotation marks for any string that might be displayed to the user.

---

### X++ Sample

This X++ code sample is in the form of a job. There is a node titled Jobs in the Application Object Tree (AOT). This sample can be added under the Jobs node, and then the job can be run.
static void JobRs001a_HelloWorld(Args _args)
{
    if (1 == 1)
    {
        // These two info() calls are identical to the X++ compiler.
        // The second form is the one typically used in X++.
        Global::info("Hello World, 1.");
        info("Hello World, 2.");
    }
    if (1 != 1)
    {
        error("This message will not appear.");
    }
    else
    {
        // These two methods are also from the Global class.
        // The + operator concatenates two strings.
        warning("This is like info, but is for warnings, 3.");
        error("This is like info, but is for errors, 4.");
    }
}

Here is the output from the Infolog window: Message (09:49:48) Hello World, 1. Hello World, 2. This is like info, but is for warnings, 3. This is like info, but is for errors, 4.

C# Sample
The following C# program is a rewrite of the previous X++ program.

using System;
class Pgm_CSharp
{
    static void Main( string[] args )
    {
        new Pgm_CSharp().Rs001a_CSharp_HelloWorld();
    }
    void Rs001a_CSharp_HelloWorld()
    {
        if (1 == 1)
        {
            Console.Out.WriteLine("Hello World, Explicit .Out , 1.");
            Console.WriteLine("Hello World, Implicit default to .Out , 2.");
        }
        if (1 != 1)
        {
            Console.Error.WriteLine("This message will not appear.");
        }
        else
        {
            Console.Error.WriteLine(".Error is like .Out, but can be for warnings, 3.");
            Console.Error.WriteLine(".Error is like .Out, but is for errors, 4.");
        }
    }
}

Here is the actual output to the C# console:

Hello World, Explicit .Out, 1.
Hello World, Implicit default to .Out, 2.
.Error is like .Out, but can be for warnings, 3.
.Error is like .Out, but is for errors, 4.
## X++, C# Comparison: Loops

This section compares the loop features between X++ and C#.

### Similarities

The following features are the same in X++ and C#:

- Declarations for variables of the int primitive data type. Declarations for other primitive types are almost the same, but the types might have different names.
- while statement for loops.
- break statement to exit a loop.
- continue statement to jump up to the top of a loop.
- \( \leq \) (less than or equal) comparison operator.

### Differences

The following table lists X++ features that are different in C#.

<table>
<thead>
<tr>
<th>FEATURES</th>
<th>X++</th>
<th>C#</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>for statement</td>
<td>The for statement is available for loops.</td>
<td>The C# for statement is slightly different from for in X++.</td>
<td>In C# you can declare the counter integer in the for statement. But in X++ the counter must be declared outside the for statement.</td>
</tr>
</tbody>
</table>
| ++ increment operator. | An ++ increment operator is available in X++. But an int variable that is decorated with ++ can only be used as a statement, not as an expression. For example, the following lines of X++ code would not compile: int age=42;
print age++;
However, the following lines of X++ code would compile: int age=42;
age++; print age; | The C# ++ operator is more flexible than in X++. | The following lines of code are the same in both languages:  
++ myInteger;  
myInteger++;  
But the following lines of code have a different effect from each other, and are valid only in C#:  
yourInt = ++myInt;  
yourInt = myInt++; |
| modulo operator. | In X++ the modulo operator is mod. | In C# the modulo operator is %. | The symbols for the modulo operator are different, but their behavior is the same in both languages. |
| Temporarily suspend a console program that has already begun. | The pause statement. | In C#, a command line program can be paused by the following line of code:  
Console.In.Read(); | In X++ you continue by clicking an OK button on a modal dialog box. In C# you continue by pressing any keyboard on the keyboard. |
In X++, the `print` statement displays a message in the Print window.

In C#, a message can be displayed on the console by the following line of code:
```csharp
Console.WriteLine();
```

The X++ `print` function is used only when you test. An X++ program that uses `print` almost always uses the `pause` statement somewhere later in the code. For production X++ code, use the Global::info Method instead of `print`. The `strfmt` function is often used together with `info`. There is no reason to use `pause` after `info`.

In X++, the `beep` function makes a sound that you can hear.

In C#, a sound that you can hear is issued by the following line of code:
```csharp
Console.Beep();
```

The statements each produce a short tone.

### Print and Global::info

The X++ code samples for loops use the `print` function to display results. In X++ you can use the `print` statement can display any primitive data type without having to call functions that convert it to a string first. This makes `print` useful in quick test situations. Generally the Global::info method is used more often than `print`. The `info` method can only display strings. Therefore the `strfmt` function is often used together with `info`. A limitation of `print` is that you cannot copy the contents of the Print window to the clipboard (such as with Ctrl+C). Global::info writes to the Infolog window which does support copy to the clipboard.

### Example 1: The while Loop

The `while` keyword supports looping in both X++ and C#.

**X++ Sample of while**

```x++
static void JobRs002a_LoopsWhile(Args _args)
{
    int nLoops = 1;
    while (nLoops <= 88)
    {
        print nLoops;
        pause;
        // The X++ modulo operator is mod.
        if ((nLoops mod 4) == 0)
        {
            break;
        }
        ++ nLoops;
    }
    beep(); // Function.
    pause; // X++ keyword.
}
```

### Output

The output in the X++ Print window is as follows:
1
2
3
4

C# Sample of while
using System;
public class Pgm_CSharp
{
static void Main( string[] args )
{
new Pgm_CSharp().WhileLoops();
}
void WhileLoops()
{
int nLoops = 1;
while (nLoops <= 88)
{
Console.Out.WriteLine(nLoops.ToString());
Console.Out.WriteLine("(Press any key to resume.)");
// Paused until user presses a key.
Console.In.Read();
if ((nLoops % 4) == 0) {
break;
}
++ nLoops;
}
Console.Beep();
Console.In.Read();
}
}
Output

The console output from the C# program is as follows:
1
(Press
2
(Press
3
(Press
4
(Press

any key to resume.)
any key to resume.)
any key to resume.)
any key to resume.)

Example 2: The for Loop
The for keyword supports looping in both X++ and C#.
X++ Sample of for

In X++ the counter variable cannot be declared as part of the for statement.


static void JobRs02a_LoopsWhileFor(Args _args)
{
    int ii; // The counter.
    for (ii=1; ii < 5; ii++)
    {
        print ii;
        pause;
        // You must click the OK button to proceed beyond a pause statement.
        // ii is always less than 99.
        if (ii < 99)
        {
            continue;
        }
        print "This message never appears.";
    }
    pause;
}

Output

The output in the X++ Print window is as follows:

1
2
3
4

C# Sample of for

using System;
public class Pgm_CSharp
{
    static void Main( string[] args )
    {
        new Pgm_CSharp().ForLoops();
    }
    void ForLoops()
    {
        int nLoops = 1, ii;
        for (ii = 1; ii < 5; ii++)
        {
            Console.Out.WriteLine(ii.ToString());
            Console.Out.WriteLine("(Press any key to resume.)");
            Console.In.Read();
            if (ii < 99)
            {
                continue;
            }
            Console.Out.WriteLine("This message never appears.");
        }
        Console.Out.WriteLine("(Press any key to resume.)");
        Console.In.Read();
    }
}

Output

The console output from the C# program is as follows:
X++, C# Comparison: Switch

In both X++ and C#, the switch statement involves the keywords case, break, and default. The following table lists the differences in the switch statement between X++ and C#.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>X++</th>
<th>C#</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>break; at the end of each case block</td>
<td>In X++, when any case block matches the expression value on the switch clause, all other case and default blocks are executed until a break; statement is reached. No break; statement is ever required in an X++ switch statement, but break; statements are important in almost all practical situations.</td>
<td>In C#, a break; statement is always needed after the statements in a case or default block. If a case clause has no statements between itself and the next case clause, a break; statement is not required between the two case clauses.</td>
<td>We recommend against omitting the break; statement after any case block, because it can confuse the next programmer who edits the code.</td>
</tr>
<tr>
<td>break; at the end of the default block</td>
<td>In X++ there is no effect of adding a break; statement at the end of the default block.</td>
<td>In C# the compiler requires a break; statement at the end of the default block.</td>
<td>For more information, see Switch Statements.</td>
</tr>
<tr>
<td>Only constant values on a case block</td>
<td>In X++ you can specify either a literal value or a variable on a case block. For example, you can write case myInteger:.</td>
<td>In C# you must specify exactly one literal value on each case block, and no variables are allowed.</td>
<td>No comments.</td>
</tr>
<tr>
<td>Multiple values on one case block</td>
<td>In X++ you can specify multiple values on each case block. The values must be separated by a comma. For example, you can write case 4,5,myInteger:.</td>
<td>In C# you must specify exactly one value on each case block.</td>
<td>In X++ it is better to write multiple values on one case block than to omit the break; statement at the end of one or more case blocks.</td>
</tr>
</tbody>
</table>

Code Examples for switch

The following sections show comparable switch statements in X++ and C#.

**X++ switch Example**

The X++ switch example shows the following:

- case iTemp: and case (93-90): to show that case expressions are not limited to constants, as they are in C#.
- //break; to show that break; statements are not required in X++, although they are almost always
static void GXppSwitchJob21(Args _args) // X++ job in AOT &gt; Jobs.
{
    int iEnum = 3;
    int iTemp = 6;
    switch (iEnum)
    {
        case 1:
            info(strFmt("iEnum is one of these values: 1,6: %1", iEnum));
            break;
        case 2, (93-90), str2Int("5"): // Equivalent to three 'case' clauses stacked, valid in X++.
            //case 2:
            //case (93-90): // Value after each 'case' can be a constant, variable, or expression; in X++.
            //case str2Int("5"):  
            info(strFmt("iEnum is one of these values: 2,3,5: %1", iEnum));
            //break; // Not required in X++, but usually wanted.
        case 4:
            info(strFmt("iEnum is one of these values: 4: %1", iEnum));
            break;
        default:
            info(strFmt("iEnum is an unforeseen value: %1", iEnum));
            break;
            // None of these 'break' occurrences in this example are required for X++ compiler.
    }
    return;
}

/*** Copied from the Infolog:
Message (02:32:08 pm)
iEnum is one of these values: 2,3,5: 3
iEnum is one of these values: 4: 3
***

C# switch Example
The C# switch example shows the following:

- **case 2, (93-90), 5:** to show that multiple expressions can be listed on one case clause in X++.

```csharp
static void GXppSwitchJob21(Args _args) // X++ job in AOT &gt; Jobs.
{
    int iEnum = 3;
    int iTemp = 6;
    switch (iEnum)
    {
        case 1:
            info(strFmt("iEnum is one of these values: 1,6: %1", iEnum));
            break;
        case 2, (93-90), 5:
            info(strFmt("iEnum is one of these values: 2,3,5: %1", iEnum));
            //break; // Not required in X++, but usually wanted.
        case 4:
            info(strFmt("iEnum is one of these values: 4: %1", iEnum));
            break;
        default:
            info(strFmt("iEnum is an unforeseen value: %1", iEnum));
            break;
            // None of these 'break' occurrences in this example are required for X++ compiler.
    }
    return;
}
```

**C# switch Example**
The C# switch example shows the following:

- **case 1:** has a comment explaining that only constant expressions can be given on a case clause.
- **break:** statements occur after the last statement in each case block that has statements, as is required by C#.
using System;
namespace CSharpSwitch2
{
    class Program
    {
        static void Main(string[] args)  // C#
        {
            int iEnum = 3;
            switch (iEnum)
            {
                case 1:  // Value after each 'case' must be a constant.
                    case 6:
                        Console.WriteLine("iEnum is one of these values: 1,6: " + iEnum.ToString());
                        break;
                //case 2,3,5:  // In C# this syntax is invalid, and multiple 'case' clauses are needed.
                        case 2:
                        case 3:
                        case 5:
                            Console.WriteLine("iEnum is one of these values: 2,3,5: " + iEnum.ToString());
                            break;
                case 4:
                    Console.WriteLine("iEnum is one of these values: 4: " + iEnum.ToString());
                    break;
                default:
                    Console.WriteLine("iEnum is an unforeseen value: " + iEnum.ToString());
                    break;
            // All 'break' occurrences in this example are required for C# compiler.
            } 
            return;
        }
    }
}

/*** Output copied from the console:
>> CSharpSwitch2.exe
iEnum is one of these values: 2,3,5: 3
>>
***/

X++, C# Comparison: String Case and Delimiters

This section compares the treatment of strings with mixed casing in X++ and C#. It also explains the string delimiters that are available in X++.

**Similarities**

The following X++ features are the same as in C#:

- The backslash (\) is the escape operator for string delimiters.
- The at sign (@) nullifies the escape effect of the backslash when the at sign is written immediately before the open quotation mark of a string.
- The plus sign (+) is the string concatenation operator.

**Differences**

X++ features that are different in C# are listed in the following table.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>X++</th>
<th>C#</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>== comparison operator</td>
<td>Insensitive: the == operator is insensitive to differences in string casing.</td>
<td>In C#, the == operator is sensitive to differences in string casing.</td>
<td>In X++ you can use the strCmp Function for case sensitive comparisons between strings.</td>
</tr>
</tbody>
</table>
In X++ you can use either the single (') or double (" ") quotation mark as the string delimiter.

**Note:** Usually the best practice is to use double quotation marks for strings that might be displayed to the user. However, it is convenient to delimit a string with single quotation marks when a double quotation mark is one of the characters in the string.

In C# you must use the double quotation mark as the string delimiter. This refers to the type `System.String`.

In X++ and C# you have the option of embedding a delimiter in a literal string and escaping it with .

In X++ you also have the alternative of embedding single quotation marks in a string that is delimited by double quotation marks (or the reverse), without having to use the escape.

---

**Character delimiters**

X++ has a string data type (str), but no character type.

In C# you must use the single quotation mark as the character delimiter. This refers to the type `System.Char`.

In the .NET Framework, a `System.String` of length one is a different data type than a `System.Char` character.

---

**Example 1: Case Sensitivity of the == Operator**

The == and != operators are case insensitive in X++, but are case sensitive in C#, as is illustrated by the following example.

<table>
<thead>
<tr>
<th>X++</th>
<th>C#</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;HELLO&quot; == &quot;hello&quot;</td>
<td>&quot;HELLO&quot; == &quot;hello&quot;</td>
<td>Different case comparisons between X++ and C#.</td>
</tr>
<tr>
<td>True in X++.</td>
<td>False in C#.</td>
<td></td>
</tr>
</tbody>
</table>

**Example 2: The + String Concatenation Operator**

The + and += operators are used to concatenate strings in both X++ and C#, as is shown by the examples in the following table.

<table>
<thead>
<tr>
<th>X++</th>
<th>C#</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>myString1 = Hello + &quot; world&quot;</code></td>
<td>(Same as for X++)</td>
<td>In both X++ and C#, the behavior of the + operator depends on the data type of its operands. The operator concatenates strings, or adds numbers.</td>
</tr>
<tr>
<td>Result is equality:</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>myString1 == &quot;Hello world&quot;</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| `myString2 = "Hello";`                        | (Same as for X++)               | In both X++ and C#, the following statements are equivalent: |
| `myString2 += " world";`                      |                                 | `a = a + b;` |
| Result is equality:                           |                                 | `a += b;` |
| `myString2 == "Hello world"`                 |                                 |                              |

**Example 3: Embedding and Escaping String Delimiters**

Either single or double quotation marks can be used to delimit strings in X++. The escape character (\) can be used to embed delimiters in a string. These are illustrated in the following table.
### Example 4: Single Escape Character

Examples that illustrate the single escape character in either the input or the output are shown in the following table.

<table>
<thead>
<tr>
<th>X++</th>
<th>C#</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>myString1 = &quot;Red\ shoe&quot;;</code></td>
<td><code>char myChar4 = 'C';</code></td>
<td>X++ has no data type that corresponds to <code>System.Char</code> in the .NET Framework. An X++ string that is limited to a length of one is still a string, not a character data type.</td>
</tr>
<tr>
<td>Result: <code>Red shoe</code></td>
<td>Here the single quotation mark is a <code>System.String</code> delimiter, not a <code>System.Char</code> delimiter.</td>
<td></td>
</tr>
<tr>
<td><code>myString2 = &quot;Red\&quot; shoe&quot;;</code></td>
<td>A literal string in C# cannot contain the two character sequence of escape followed by a space, such as <code>\ &quot;</code>. A compiler error occurs.</td>
<td>When the X++ compiler encounters the two character sequence of <code>\ </code>, it discards the single escape character.</td>
</tr>
<tr>
<td>Result: <code>Red\ shoe</code></td>
<td>(Same as for X++.)</td>
<td>In a pair of escape characters, the first negates the special meaning of the second.</td>
</tr>
</tbody>
</table>

### Comparison: Array Syntax

There are similarities and differences in the features and syntax for arrays in X++ versus C#.

#### Similarities

Overall there is much similarity in the syntax and treatment of arrays in X++ and C#. However there are many differences.

#### Differences

The following table lists areas in the `[]` syntax for arrays that are different for X++ and C#.
<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>X++</th>
<th>C#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td>An array is declared with square brackets appended to the variable name.</td>
<td>An array is declared with square brackets appended to the data type.</td>
</tr>
<tr>
<td></td>
<td>Note: An X++ array cannot be a parameter in a method.</td>
<td>int myInts[]; // C#</td>
</tr>
<tr>
<td>Declaration</td>
<td>The array syntax supports only primitive data types, such as int and str. The syntax does not support classes or tables.</td>
<td>In X++ you can use the Array class for an array of objects.</td>
</tr>
<tr>
<td></td>
<td>In X++ you cannot have an array of arrays. However, there is advanced syntax for limiting the amount of active memory that a large array can consume, which looks like the multi-dimensional syntax in C#: int intArray[1024,16];. For more information, see Best Practice Performance Optimizations: Swapping Arrays to Disk.</td>
<td></td>
</tr>
<tr>
<td>Declaration</td>
<td>X++ is limited to single dimension arrays (myStrings[8]).</td>
<td>C# adds support for multi-dimensional arrays (myStrings[8,3]) and for jagged arrays (myStrings[8] [3]).</td>
</tr>
<tr>
<td>Declaration</td>
<td>In X++ an array is a special construct but it is not an object.</td>
<td>In C# all arrays are objects regardless of syntax variations.</td>
</tr>
<tr>
<td>Length</td>
<td>In X++ the length of a static sized array is determined in the declaration syntax.</td>
<td>In C# the size of an array is determined when the array object is constructed.</td>
</tr>
<tr>
<td>Length</td>
<td>An X++ array can have a dynamic length that can be increased even after population has begun. This applies only when the array is declared without a number inside the[]. Performance might be slowed if the length of the dynamic array is increased many times.</td>
<td>When you use the [] declaration syntax in X++, no more preparation is needed before you assign values to the array. In C# you must declare and then construct the array before assigning to it.</td>
</tr>
<tr>
<td>Length</td>
<td>In the following fragment of X++ code, only the myInts array is dynamic and can increase in size.</td>
<td></td>
</tr>
</tbody>
</table>

```csharp
int myInts[]; // X++
int myBools[5];
myInts[2] = 12;
myInts[3] = 13;
myBools[6] = 26; // Error
```
Length

You can get the length of some arrays by using the `dimOf` function. C# arrays are objects that have a `Length` property. No comments.

Indexing

Array indexing is 1 based. Array indexing is 0 based. `mtIntArray[0]` would cause an error in X++.

Constant

In X++ a constant value is best achieved by using the `#define` precompiler directive. In C# you can decorate your variable declaration with the keyword `const`, to achieve a constant value. X++ has no `const` keyword. C# cannot assign values to variables that are created by its `#define` precompiler directive.

X++ and C# Samples

The following code samples show how arrays of primitive data types are handled. The first sample is in X++, and the second sample is in C#. Both samples achieve the same results.

X++ Sample

```x++
static void JobRs005a_ArraySimple(Args _args)
{
    #define.macroArrayLength(3)
    // Static length.
    str sSports[3];
    // Dynamic length, changeable during run time.
    int years[];
    int xx;
    Global::warning("-------- SPORTS --------");
    sSports[0] = "Baseball";
    for (xx=1; xx <= 3; xx++)
    {
        info(int2str(xx) + " , [" + sSports[xx] + "]");
    }
    warning("-------- YEARS --------");
    years[4] = 2008;
    years[10] = 1930;
    for (xx=1; xx <= 10; xx++)
    {
        info(int2str(xx) + " , " + int2str(years[xx]));
    }
}
```

Output

The output to the Infolog is as follows:
using System;
public class Pgm_CSharp
{
    static public void Main( string[] args )
    {
        new Pgm_CSharp().ArraySimple();
    }
    private void ArraySimple()
    {
        const int const_iMacroArrayLength = 3;
        // In C# the length is set at construction during run.
        string[] sSports;
        int[] years;
        int xx;
        Console.WriteLine("-------- SPORTS --------");
        sSports = new string[const_iMacroArrayLength];
        sSports[const_iMacroArrayLength - 1] = "Baseball";
        for (xx=0; xx < const_iMacroArrayLength; xx++)
        {
        }
        Console.WriteLine("-------- YEARS --------");
        // In C# you must construct the array before assigning to it.
        years = new int[10];
        years[4] = 2008;
        years[10 - 1] = 1930;
        for (xx=0; xx < 10; xx++)
        {
            Console.WriteLine(xx.ToString() + ", [" + years[xx].ToString() + "]");
        }
    }
} // EOClass

The output from the C# program to the command line console is as follows:
Additional array-like X++ features

The **container** is a special data type that is available in X++. It can be considered as similar to an array, or similar to a **List** collection.

**Comparison: Collections**

In a Finance and Operations application, you can use the X++ **List** collection class. The .NET Framework that is used in C# has a similar class named `System.Collections.Generic.List`.

**Comparing the Use of the List Classes**

The following table compares methods on the X++ **List** class to the methods on `System.Collections.Generic.List` from the .NET Framework and C#.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>X++</th>
<th>C#</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration of collection</td>
<td><code>List myList;</code></td>
<td><code>List&lt;string&gt; myList;</code></td>
<td>The X++ declaration does not include the type of elements to be stored.</td>
</tr>
<tr>
<td>Declaration of iterator</td>
<td><code>ListIterator iter</code></td>
<td><code>IEnumerator&lt;string&gt; iter;</code></td>
<td>In X++ the <code>ListIterator</code> object has methods that can insert and delete items from the <code>List</code>. The X++ <code>ListEnumerator</code> object cannot modify the contents of the <code>List</code>. In X++ the <code>ListEnumerator</code> object is always created on the same tier as the <code>List</code>. This is not always true for <code>ListIterator</code>.</td>
</tr>
<tr>
<td>Obtaining an iterator</td>
<td><code>new ListIterator (myList)</code></td>
<td><code>myList.GetEnumerator()</code></td>
<td>In both X++ and C#, the List object has a getter method for an associated enumerator.</td>
</tr>
</tbody>
</table>
Constructor

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>X++</th>
<th>C#</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>new List(Types::String)</td>
<td>new List&lt;string&gt;()</td>
<td>Information about the type of objects to be stored inside the List classes is given to the constructor in both X++ and C#.</td>
</tr>
</tbody>
</table>

Updating data

<table>
<thead>
<tr>
<th>Updating data</th>
<th>X++</th>
<th>C#</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Enumerator – the enumerator becomes invalid if any items in the List are added or removed. Iterator – the iterator has methods that insert and delete items from the List. The iterator remains valid.</td>
<td>Enumerator – the enumerator becomes invalid if any items in the List are added or removed.</td>
<td>Enumerators become invalid after items are added or deleted from the List, in both X++ and C#.</td>
</tr>
</tbody>
</table>

Update data

<table>
<thead>
<tr>
<th>Updating data</th>
<th>X++</th>
<th>C#</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In X++ the List class has methods for adding items at the start or end of the list.</td>
<td>In C# the List class has methods for adding members at any position in the list. It also has methods for removing items from any position.</td>
<td>In X++ items can be removed from the List only by an iterator.</td>
</tr>
</tbody>
</table>

Example 1: Declaration of a List

Following are code examples in X++ and C# that declare List collections.

```cpp
// X++
List listStrings, list2, listMerged;
ListIterator iterator;
```

```csharp
// C#
using System;
using System.Collections.Generic;
List<string> listStrings, list2, listMerged; IEnumerator<string> iterator;
```

Example 2: Construction of a List

In both languages, the type of items that the collection stores must be specified at the time of construction. For class types, X++ can get no more specific than whether the type is a class (Types::Class). Following are code examples in X++ and C#.

```cpp
// X++
listStrings = new List(Types::String);
```

```csharp
// C#
listStrings = new List<string>;
```

Example 3: Add Items to a List

In both X++ and C#, the collection provides a method for appending an item to the end of the collection, and for inserting an item the start. In C# the collection provides a method for inserting at any point in the collection based on an index value. In X++ a collection iterator can insert an item at its current position. Following are code
examples in X++ and C#.

```csharp
// X++
listStrings.addEnd ("StringBB.");
listStrings.addStart ("StringAA.");
// Iterator performs a midpoint insert at current position.
listIterator.insert ("dog");
```

```csharp
// C#
listStrings.Add ("StringBB.");
listStrings.Insert (0,"StringAA.");
// Index 7 determines the insertion point.
listStrings.Insert (7,"dog");
```

**Example 4: Iterate Through a List**

Both X++ and C# have iterator classes that you can use to step through the items in a collection as shown in the following examples.

```csharp
// X++
literator = new ListIterator (listStrings);
// Now the iterator points at the first item.

// The more method answers whether
// the iterator currently points
// at an item.
while (literator.more())
{
    info(any2str (literator.value()));
    iterator.next();
}
```

```csharp
// C#
literator = listStrings .GetEnumerator();
// Now enumerator points before the first item, not at the first item.

// The MoveNext method both advances the item pointer, and
// answers whether the pointer is pointing at an item.
while (literator.MoveNext())
{
    Console.WriteLine (literator.Current);
}
```

**Example 4b: foreach in C#**

In C# the `foreach` keyword is often used to simplify the task of iterating through a list. The following code example behaves the same as the previous C# example.

```csharp
foreach (string currentString in listStrings)
{
    Console.WriteLine(currentString);
}
```

**Example 5: Delete the Second Item**

The following code examples delete the second item from the collection. In X++ this requires an iterator. In C# the collection itself provides the method for removing an item.
Example 6: Combine Two Collections

The following code examples combine the contents of two collections into one.

```
// X++
iterator.begin();
iterator.next();
iterator.delete();

// C#
listStrings.RemoveAt(1);
```

```
// X++
listStrings = List::merge(listStrings ,listStr3);
// Or use the .appendList method:
listStrings.appendList (listStr3);

// C#
listStrings.InsertRange(listStrings.Count ,listStr3);
```

Comparison: Collections of keys with values

In a Finance and Operations application, you can use the `Map` collection class. The `Map` collection holds pairs of values, the key value plus a data value. This resembles the .NET Framework class named `System.Collections.Generic.Dictionary`.

The following list describes similarities between X++ and C# regarding their collections that store key-value pairs:

- Both prevent duplicate keys.
- Both use an enumerator (or iterator) to loop through the items.
- Both key-value collection objects are constructed with designations of the types that are stored as key and value.
- Both can store class objects, and are not limited to storing primitives like `int`.

The following table describes differences between X++ and C# regarding their collections classes that store key-value pairs:

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>X++</th>
<th>C#</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duplicate keys</td>
<td>In X++ the <code>Map</code> class prevents duplicate keys by implicitly treating your call to its <code>insert</code> method as an operation to update only the value associated with the key.</td>
<td>In C# the <code>Dictionary</code> class throws an exception when you try to add a duplicate key.</td>
<td>Duplicate keys are prevented in both languages, although by different techniques.</td>
</tr>
</tbody>
</table>
**Delete items**

In X++ the `delete` method on an iterator object is used to remove an unwanted key-value pair from a `Map`

In C# the `Dictionary` class has a `remove` method.

In both languages, an enumerator is made invalid if the collection item count is modified during the life of the enumerator.

---

### Example 1: Declaration of a Key-Value Collection

In both languages, the type of items that the key-value collection stores must be specified. In X++ the type is specified at time of construction. In C# the type is specified at both the time of declaration and the time of construction. Following are code examples in X++ and C#.

```csharp
// X++
Map mapKeyValue;
MapEnumerator enumerator;
MapIterator mapIterator;

// C#
Dictionary<int,string> dictKeyValue;
IEnumerator<SysCollGen.KeyValuePair<int,string>> enumer;
KeyValuePair<int,string> kvpCurrentKeyValuePair;
```

### Example 2: Construction of the Collection

In both languages, the type of items that the key-value collection stores specified during construction. For class types, X++ can get no more specific than whether the type is a class (Types::Class). Following are code examples in X++ and C#.

```csharp
// X++
mapKeyValue = new Map(Types::Integer, Types::String);

// C#
dictKeyValue = new Dictionary<int,string>();
```

### Example 3: Add an Item to the Collection

There is almost no difference in how an item is added to a key-value collection in X++ and C# as shown in the following code examples.

```csharp
// X++
mapKeyValue.insert(xx ,int2str(xx) + "_Value");

// C#
dictKeyValue.Add(xx ,xx.ToString() + "_Value");
```

### Example 4: Iterate Through a Key-Value Collection

Enumerators are used to loop through the key-value collections in both X++ and C# as shown in the following code examples.
Example 5: Update the Value Associated with a Key

The syntax is very different between the two languages for an update of the value associated to a given key. Following are code examples for the key 102.

```cpp
// X++
mapKeyValue.insert(
    102,
    ".insert(), Re-inserted" + " key 102 with a different value.";)
```

```csharp
// C#
dictKeyValue[102] =
    "The semi-hidden .item property in C#, Updated the value for key 102.";
```

Example 6: Delete One Item

The syntax is very different between the two languages to delete one key-value pair from a collection, while iterating through the collection members. Code examples for the key 102 are shown below.

```cpp
// X++
mapIter = new MapIterator(mapKeyValue);
//mapIter.begin();
while (mapIter.more())
{
    iCurrentKey = mapIter.key();
    if (104 == iCurrentKey)
    {
        // mapKeyValue.remove would invalidate the iterator.
        mapIter.delete();
        break;
    }
    mapIter.next();
}
```

```csharp
// C#
dictKeyValue.Remove(104);
```

Comparison: Exceptions
There are some similarities but many differences when we compare exception related behavior between X++ and C#. The **try**, **catch**, and **throw** keywords behave the same in X++ and C#. But the types of exceptions thrown and caught are different for the two languages.

**Similarities**

Similarities between X++ and C# regarding their exception features include the following:

- Both languages have the same **try** keyword.
- Both have the same **catch** keyword.
- Both enable for a **catch** statement that does not specify any particular exception. Such a **catch** statement catches all exceptions that reach it.
- Both have the same **throw** keyword.

**Differences**

Exception-related differences between X++ and C# are described in the following table.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>X++</th>
<th>C#</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>retry</td>
<td>Jumps to the first instruction in the associated <strong>try</strong> block. For more information, see Exception Handling with try and catch Keywords.</td>
<td>The functionality of the <strong>retry</strong> keyword can be mimicked in C# code, but there is no corresponding keyword.</td>
<td>Only X++ has a <strong>retry</strong> keyword. C# has no counterpart. For more information, see X++, C# Comparison: Automated Retry After an Exception.</td>
</tr>
<tr>
<td>finally</td>
<td>The <strong>finally</strong> keyword is supported to follow the <strong>try</strong> and <strong>catch</strong> keywords.</td>
<td>The <strong>finally</strong> keyword marks a block of code that follows the <strong>try</strong> and <strong>catch</strong> blocks. The finally will be executed regardless of whether any exception is thrown or caught.</td>
<td>The semantics are identical to the semantics in C#.</td>
</tr>
<tr>
<td>Specific exceptions</td>
<td>In X++ an exception is an element of the <strong>Exception</strong> enum, such as <strong>Error</strong>, <strong>Deadlock</strong>, or <strong>CodeAccessSecurity</strong>. No exception can contain another.</td>
<td>In C# an exception is an instance of the <strong>System.Exception</strong> base class, or any class that inherits from it. An exception can be contained in the <strong>InnerException</strong> property of the thrown exception.</td>
<td>In X++ each thrown exception is a value of the Exception enum. For more information, see Exception Enumeration.</td>
</tr>
<tr>
<td>Exception message</td>
<td>In X++ the message that is created when an exception is raised is available only in the Infolog, and the message is not directly tied to the exception.</td>
<td>In C# the message is the <strong>Message</strong> member of the <strong>System.Exception</strong> object.</td>
<td>In X++ the Global::error method is the mechanism that display exception messages in the Infolog. For more information, see Exception Handling with try and catch Keywords.</td>
</tr>
</tbody>
</table>
Exception conditions

In X++ an error occurs when you call an instance method on an object variable that has not yet had anything assigned to it. However, no exception is raised along with this error. Therefore no catch block can gain control even if the unassigned variable is misused in a try block. In the following code example, the error caused by the code: `DialogBox box4; try { box4.toString(); info("toString did not error, but expected an error."); } catch (Exception::Error) // No Exception value catches this. { info("Invalid use of box4 gave control to catch, unexpected."); }` does not cause control to transfer to any catch block.

In C# a `System.NullReferenceException` is raised when an uninitialized variable is treated as an object reference. There might be several other differences in the conditions that raise exceptions.

SQL transactions

In X++ when an SQL exception occurs in a `ttsBegin - ttsCommit` transaction, no catch statement inside the transaction block can process the exception.

In C# a catch block inside an SQL transaction can catch the exception.

Examples

The following X++ features are demonstrated:

- try keyword.
- catch keyword.
- The behavior after an Exception::Error exception occurs.

X++ Example
static void JobRs008a_Exceptions(Args _args)
{
    str sStrings[4];
    int iIndex = 77;
    try
    {
        info("On purpose, this uses an invalid index for this array: " + sStrings[iIndex]);
        warning("This message does not appear in the Infolog," + " it is unreached code.");
    }
    // Next is a catch for some of the values of
    // the X++ Exception enumeration.
    catch (Exception::CodeAccessSecurity)
    {
        info("In catch block for -- Exception::CodeAccessSecurity");
    }
    catch (Exception::Error)
    {
        info("In catch block for -- Exception::Error");
    }
    catch (Exception::Warning)
    {
        info("In catch block for -- Exception::Warning");
    }
    catch
    {
        info("This last 'catch' is of an unspecified exception.");
    }
    //finally
    //{
    //    //Global::Warning("'finally' is not an X++ keyword, although it is in C#.");
    //}
    info("End of program.");
}

Here is the output from the Infolog window:

Message (18:07:24)
Error executing code: Array index 77 is out of bounds.
Stack trace
(C)\Jobs\JobRs008a_Exceptions - line 8
In catch block for -- Exception::Error
End of program.

C# Sample
The following C# program is a rewrite of the previous X++ program.
Here is the output to the C# console:

(e2) In catch block for -- System.IndexOutOfRangeException
'finally' is not an X++ keyword, although it is in C#.
End of program.

Comparison: Automated Retry After an Exception

Sometimes you can write code in a catch block that fixes the cause of an exception that occurs during run time. X++ provides a **retry** keyword that can be used only inside a **catch** block. The **retry** keyword enables a program to jump back to the start of the **try** block after the problem has been corrected by code in the **catch** block. C# does not have a **retry** keyword. However, C# code can be written to provide equivalent behavior.
Code Samples for Retry

The following X++ sample program causes an Exception::Error to be raised. This occurs when it first tries to read an element from the `sStrings` array by using an invalid index value. When the exception is caught, corrective action is taken during run time inside the `catch` block. The retry statement then jumps back to the first statement in the `try` block. This second iteration works without encountering any exception.

```csharp
static void JobRs000b_ExceptionsAndRetry(Args _args)
{
    str sStrings[4];
    str sTemp;
    int iIndex = 0;

    sStrings[1] = "First array element.";
    try
    {
        print("At top of try block: " + int2str(iIndex));
        sTemp = sStrings[iIndex];
        print("The array element is: " + sTemp);
    }
    catch (Exception::Error)
    {
        print("In catch of -- Exception::Error (will retry)." + " Entering catch.");
        ++iIndex;
        print("In catch of -- Exception::Error (will retry)." + " Leaving catch.");
        // Here is the retry statement.
        retry;
    }
    print("End of X++ retry program.");
    pause;
}
```

Output

Here is the output to the Print window:

```
At top of try block: 0
In catch of -- Exception::Error (will retry). Entering catch.
In catch of -- Exception::Error (will retry). Leaving catch.
At top of try block: 1
The array element is: First array element.
End of X++ retry program.
```

C# Sample

The following C# sample is not a line-by-line translation from the previous X++ sample. Instead the C# program has a different structure so that it mimics the behavior of the `retry` keyword that the X++ program relies on. The `try` and `catch` blocks are in a called method. The variables that are used in the `try` block are stored in the caller method. The caller method passes the variables as parameters that are decorated with the `ref` keyword, so that their values can be corrected inside the `catch` block of the called method. The called method captures all exceptions, and returns a `boolean` to communicate back to the caller whether a second call is required.
using System;
public class Pgm_CSharp
{
    static void Main(string[] args)
    {
        new Pgm_CSharp().Rs008b_CSharp_ExceptionsAndRetry();
    }
    void Rs008b_CSharp_ExceptionsAndRetry() // Caller
    {
        int iIndex = -1,
            iNumRetriesAllowed = 3;
        bool bReturnCode = true; // Means call the callee method.
        for (int xx=0; xx <= iNumRetriesAllowed; xx++)
        {
            if (bReturnCode)
            {
                bReturnCode = this.Rs008b_CSharp_ExceptionsAndRetry_Callee(ref iIndex);
            }
            else
            {
                break;
            }
        }
        Console.WriteLine("End of C# caller method.");
    }
    private bool Rs008b_CSharp_ExceptionsAndRetry_Callee(ref int iIndex)
    {
        bool bReturnCode = true; // Means call this method again.
        string[] sStrings = new string[4];
        string sTemp;
        sStrings[0] = "First array element."
        try
        {
            Console.WriteLine("At top of try block: " + iIndex.ToString());
            sTemp = sStrings[iIndex];
            Console.WriteLine("The array element is: " + sTemp);
            bReturnCode = false; // Means do not call this method again.
        }
        catch (Exception)
        {
            Console.WriteLine("In catch of -- Exception. Entering catch.");
            ++iIndex; // The 'ref' parameter in C#.
            Console.WriteLine("In catch of -- Exception. Leaving catch.");
            //retry;
            // In C# we let the caller method do the work
            // that the retry keyword does in X++.
        }
        Console.WriteLine("End of C# callee method.");
        return bReturnCode;
    }
}

Output
Here is the output to the console:
Comparison: Operators

This section compares the operators between X++ and C#.

Assignment Operators

The following table displays the differences between the assignment operators in X++ and C#.

<table>
<thead>
<tr>
<th>X++ AND C#</th>
<th>DIFFERENCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>In X++ this operator causes an implicit conversion whenever a loss of precision might occur, such for an assignment from an int64 to an int. But in C# the assignment causes a compile error.</td>
</tr>
<tr>
<td>+= and -=</td>
<td>The only difference is that in C# these operators are also used in delegate manipulation.</td>
</tr>
<tr>
<td>++ and --</td>
<td>These are the increment and decrement operators in both languages. The following line is identical in both languages: ++myInteger; But in X++ these two operators are for statements, not for expressions. Therefore the following lines generate compile errors in X++: myStr = int2str(++myInteger); myIntA = myIntBB++;</td>
</tr>
</tbody>
</table>

Arithmetic Operators

The following table lists the arithmetic operators.

<table>
<thead>
<tr>
<th>X++ AND C#</th>
<th>DIFFERENCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>As the multiplication operator, there are no differences. <strong>Note:</strong> The asterisk is also used in the SQL statements that are part of the X++ language. In these SQL statements the asterisk can also be one of the following: • A wildcard indicating that all the columns should be returned. • A wildcard for characters in a string that is used on a like clause.</td>
</tr>
<tr>
<td>/</td>
<td>The division operator is the same in X++ and C#.</td>
</tr>
<tr>
<td>MOD</td>
<td>For modulo operations, the only difference is that the % symbol is used in C#.</td>
</tr>
</tbody>
</table>
The addition operator is the same in X++ and C#. The plus sign is also used for string concatenation. This operator adds numbers and concatenates strings in both languages.

The subtraction operator is the same in X++ and C#.

### Bitwise Operators

The following table compares the bitwise operators between X++ and C#.

<table>
<thead>
<tr>
<th>X++ AND C#</th>
<th>DIFFERENCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&lt;</td>
<td>The left shift operator is the same in X++ and C#.</td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>The right shift operator is the same in X++ and C#.</td>
</tr>
<tr>
<td>~</td>
<td>The bitwise NOT operator is the same in X++ and C#.</td>
</tr>
<tr>
<td>&amp;</td>
<td>The binary AND operator is the same in X++ and C#.</td>
</tr>
<tr>
<td>^</td>
<td>The binary XOR operator is the same in X++ and C#.</td>
</tr>
</tbody>
</table>

### Relational Operators

The following relational operators are the same in X++ and C#:

- ==
- <=
- >
- <
- !=
- &&
- ||
- !
- ?? :

### Comparison: Events

There are some differences in how X++ and C# implement the event design pattern. For more information, see Event Terminology and Keywords.

### Comparison of Events between X++ and C#

There are differences in the way delegates are used for events in X++ versus C#.
<table>
<thead>
<tr>
<th>CONCEPT</th>
<th>X++</th>
<th>C#</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>delegate</td>
<td>In X++, a delegate can be declared only as a member on a class. A delegate cannot be a member on a table. All delegates are instance members of their class, not static members. No access modifier can be used on a delegate declaration, because all delegates are protected members. Therefore, the event can be raised only by code within the same class where the delegate is a member. However, the one exception to the private nature of a delegate is that code outside their class can operate on the delegates by using the += and -= operators.</td>
<td>In C#, each delegate is a type, just as every class is a type. A delegate is declared independently of any class. Without the event keyword, you can have a delegate as a parameter type on a method, just as you can have a class as a parameter type. You can construct an instance of a delegate to pass in for the parameter value.</td>
<td>In X++, each class is a type, but no delegate is a type. You cannot construct an instance of a delegate. No delegate can be a parameter for a method. But you can create a class that has a delegate member, and you can pass instances of the class as parameter values. For more information, see X++ Keywords.</td>
</tr>
<tr>
<td>event</td>
<td>In X++ code, an event is one of the following:</td>
<td>In C#, the event keyword is used to declare a delegate type as a member of a class. The effect of the event keyword is to make the delegate protected, yet still accessible for the += and -= operators. You can subscribe event handler methods to an event by using the += operator. A delegate can be useful without the event keyword, as a technique for passing a function pointer as a parameter into a method.</td>
<td>The automatic events that occur before the start of a method, and after the end of a method, can be subscribed to only by using the AOT.</td>
</tr>
<tr>
<td>+= and -= operators</td>
<td>In X++, you use the += operator to subscribe methods to a delegate. The -= operator unsubscribes a method from a delegate.</td>
<td>In C#, you use the += operator to subscribe methods to an event, or to a delegate that is not used with the event keyword.</td>
<td>The delegate contains a reference to all the objects that have methods subscribed to the delegate. Those objects are not eligible for garbage collection while delegate holds those references.</td>
</tr>
<tr>
<td>eventHandler</td>
<td>In X++, the eventHandler keyword is required when you use either the += or -= operator to subscribe or unsubscribe a method from a delegate.</td>
<td>System.EventHandler is a delegate type in the .NET Framework.</td>
<td>This term is used differently in X++ than it is in C# or the .NET Framework. For more information, see X++ Keywords.</td>
</tr>
</tbody>
</table>

**X++ Example**

The important things to notice in the X++ example are the following:
The `XppClass` has a delegate member that is named `myDelegate`.

```csharp
NOTE
The AOT contains a node for the delegate. The node is located at AOT > Classes > XppClass > myDelegate. Several event handler nodes can be located under the myDelegate node. Event handlers that are represented by nodes in the AOT cannot be removed by the -= operator during run time.
```

- The `{}` braces at the end of the delegate declaration are required, but they cannot have any code in them.
- The `XppClass` has two methods whose parameter signatures are compatible with the delegate. One method is static.
- The two compatible methods are added to the delegate with the `+=` operator and the `eventHandler` keyword. These statements do not call the event handler methods, the statements only add the methods to the delegate.
- The event is raised by one call to the delegate.
- The parameter value that passed in to the delegate is received by each event handler method.
- The short X++ job at the top of the example starts the test.

```
// X++
// Simple job to start the delegate event test.
static void DelegateEventTestJob()
{
    XppClass::runTheTest("The information from the X++ job.");
}
// The X++ class that contains the delegate and the event handlers.
class XppClass
{
    delegate void myDelegate(str _information)
    {
    }
    public void myEventSubscriberMethod2(str _information)
    {
        info("X++, hello from instance event handler 2: " + _information);
    }
    static public void myEventSubscriberMethod3(str _information)
    {
        info("X++, hello from static event handler 3: " + _information);
    }
    static public void runTheTest(str _stringFromJob)
    {
        XppClass myXppClass = new XppClass();
        // Subscribe two event handler methods to the delegate.
        myXppClass.myDelegate += eventHandler(myXppClass.myEventSubscriberMethod2);
        myXppClass.myDelegate += eventHandler(XppClass::myEventSubscriberMethod3);
        // Raise the event by calling the delegate one time,
        // which calls all the subscribed event handler methods.
        myXppClass.myDelegate(_stringFromJob);
    }
}
```

The output from the previous X++ job is as follows:

```
X++, hello from static event handler
3: The information from the X++ job. X++, hello from instance event handler
2: The information from the X++ job.
```
This section contains a C# code sample for the event design pattern of the previous X++ sample.

```csharp
// C#
using System;
// Define the delegate type named MyDelegate.
public delegate void MyDelegate(string _information);
public class CsClass
{
    protected event MyDelegate MyEvent;
    static public void Main()
    {
        CsClass myCsClass = new CsClass();
        // Subscribe two event handler methods to the delegate.
        myCsClass.MyEvent += new MyDelegate(myCsClass.MyEventSubscriberMethod2);
        myCsClass.MyEvent += new MyDelegate(CsClass.MyEventSubscriberMethod3);
        // Raise the event by calling the event one time, which
        // then calls all the subscribed event handler methods.
        myCsClass.MyEvent("The information from the C# Main.");
    }
    public void MyEventSubscriberMethod2(string _information)
    {
        Console.WriteLine("C#, hello from instance event handler 2: " + _information);
    }
    static public void MyEventSubscriberMethod3(string _information)
    {
        Console.WriteLine("C#, hello from static event handler 3: " + _information);
    }
}
```

The output from the previous C# sample is as follows:

```
CsClass.exe C#, hello from instance event handler
2: The information from the C# Main. C#, hello from static event handler
3: The information from the C# Main.
```

Events and the AOT

There are other event systems that apply only to items in the AOT. For more information, see Event Handler Nodes in the AOT.

Comparison: Precompiler Directives

X++ and C# share some keywords for their precompiler directive syntax, but the meanings are not always the same.

Similarities

The X++ and C# compilers recognize many of the same keywords. In most cases, the keywords mean the same for both language compilers.

Differences

A fundamental difference between the precompiler directives in X++ versus C# is the #define keyword that both language precompilers recognize. Unlike C#, in X++ the #define directive requires a dot in its syntax. In X++, parentheses can be used to give the defined symbol a value. These differences are shown in the following examples:

- In X++: #define InitialYear(2003)
- In C#: #define InitialYear

A minor difference is that in C# there can be spaces and tab characters between the # character and the directive
Identical Keywords

The following table lists precompiler directives that are similar in X++ and C#.

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>X++</th>
<th>C#</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>#define</td>
<td>In X++, a precompiler variable name can be defined, and a value can be given to that variable.</td>
<td>In C#, a precompiler variable name can be defined, but no value can be given to that variable. Also, any #define in C# must occur at the top of the file, and cannot occur after any code such as a using statement or a class declaration.</td>
<td>The C# compiler can input a command line parameter of `/define` to define a precompiler variable name without defining the variable in any C# code file. The X++ compiler has no counterpart to `/define`.</td>
</tr>
<tr>
<td>#if</td>
<td>In X++, #if can determine whether a precompiler variable exists, and whether the variable has a given value.</td>
<td>In C#, #if can only determine whether a precompiler variable exists. It cannot test for any value because no value can be assigned.</td>
<td></td>
</tr>
<tr>
<td>#endif</td>
<td>In X++, #endif marks the end of an #if block. It also ends an #ifnot block.</td>
<td>In C#, #endif marks the end of an #if block, regardless of whether the block includes a #else.</td>
<td></td>
</tr>
</tbody>
</table>

Different Keywords with the Same Processing Result

The following table lists precompiler directives that are named differently in X++ and C#, but that give the same results when processed.

<table>
<thead>
<tr>
<th>X++</th>
<th>C#</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>#ifnot</td>
<td>#if #else</td>
<td>There is no #else directive in X++, but the #ifnot provides similar functionality. In X++, #ifnot can determine whether a precompiler variable exists, and whether the variable does not have a specific given value. In C#, #if can determine whether a precompiler variable exists when the <code>!</code> symbol is prefixed to the variable name.</td>
</tr>
<tr>
<td>//BP Deviation documented</td>
<td>#pragma warning</td>
<td>These X++ and C# entries are not equivalent, but there is a partial similarity. Both suppress compiler warning messages.</td>
</tr>
<tr>
<td>#macrolib</td>
<td>.HPP file in C++</td>
<td>There is a partial similarity between the X++ directive #macrolib versus an .HPP file in C++. Both can contain several #define statements.</td>
</tr>
</tbody>
</table>

Precompiler Directives Exclusive to X++

The following table lists X++ precompiler directives that have no direct counterpart in C#.
### X++

<table>
<thead>
<tr>
<th>Feature</th>
<th>X++</th>
<th>C#</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>#linenumber</strong></td>
<td>The #linenumber directive is for obtaining the line number, so that it can be output to the Infolog. The C# directive #line is different because its purpose is for setting the line number.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>#defdec #definc</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>#globaldefine</strong></td>
<td>In X++, there is a small difference between #globaldefine versus #define. The difference is that #globaldefine never overwrites a current nonnull value that was assigned to a precompiler variable by #define. C# has nothing similar to this difference, because in C#, a precompiler variable name cannot be given a value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>#localmacro #macro</strong></td>
<td>In X++, #localmacro enables you to assign a multiline value to a precompiler variable. #macro is a synonym, but #localmacro is recommended. In C#, the #define directive has part of this functionality, but it cannot assign a value to a precompiler variable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>#globalmacro</strong></td>
<td>In X++, #globalmacro is almost the same as the preferred #localmacro.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Comparison: Object Oriented Programming

The object oriented programming (OOP) principles of X++ differ from C#.

**Conceptual Comparisons**

The following table compares the implementation of OOP principles between X++ and C#.

<table>
<thead>
<tr>
<th>Feature</th>
<th>X++</th>
<th>C#</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Casting</strong></td>
<td>The X++ language has the keywords is and as, which are used to make downcasts safe and explicit. Tip: X++ does not require the use of the as keyword when you downcast a base class variable to a derived class variable. However, we recommend that all downcast statements use the as keyword.</td>
<td>An object can be cast either up or down the inheritance path. Downcasts require the as keyword.</td>
<td>For more information about the X++ keywords is and as, see Expression Operators: Is and As for Inheritance.</td>
</tr>
<tr>
<td><strong>Local functions</strong></td>
<td>A method can contain a declaration and code body for zero or more local functions. Only that method can have calls to the local function.</td>
<td>C# 3.0 supports lambda expressions, which have some similarity to anonymous functions and local functions. Lambda expressions are often used with delegates.</td>
<td></td>
</tr>
<tr>
<td>FEATURE</td>
<td>X++</td>
<td>C#</td>
<td>COMMENTS</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------</td>
<td>------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Method overloading</td>
<td>Method overloading is not supported. A method name can occur only one time per class.</td>
<td>Method overloading is supported. A method name can occur multiple times in one class, with different parameter signatures in each case.</td>
<td>X++ does support optional parameters on methods. Optional parameters can partially mimic method overloading. For more information, see the row for optional parameters in this table.</td>
</tr>
<tr>
<td>Method overriding</td>
<td>Method overriding is supported. A derived class can have a method by the same name as in the base class, as long as the parameter signature is the same in both cases. The only exception is that the overriding method can add a default value to a parameter.</td>
<td>Method overriding is supported. The <code>virtual</code> keyword must be applied to a method before the method can be overridden in a derived class.</td>
<td>The concept of overriding a method includes the method name, its parameter signature, and its return type. The concept of method overriding does not apply if the base method and the overriding method differ in any of these aspects.</td>
</tr>
<tr>
<td>Optional parameters</td>
<td>A parameter declaration can be followed by a default value assignment. The method caller has the option of passing a value for that parameter, or ignoring the parameter to accept the default value. This feature mimics method overloading because two calls to the same method name can pass different numbers of parameters. Each parameter that has a default value must follow the last parameter that does not have a default value.</td>
<td>Optional parameters are supported by the <code>params</code> keyword. Even without the <code>params</code> keyword, from the point of view of the caller, method overloading can provide partially similar functionality.</td>
<td>For more information, see Parameters and Scoping and Using Optional Parameters.</td>
</tr>
</tbody>
</table>
### Single Inheritance

You can derive your X++ class from another X++ class by using the `extends` keyword in the `classDeclaration` node of your class, in the AOT. No class implicitly derives directly from another class. If you want your class to directly derive from the `Object` class, you must use the `extends` keyword. You can specify only one class on the `extends` keyword.

**Caution:** When you modify an X++ base class that other classes derive from, you must recompile that base class using the Compile forward. This option ensures that the derived classes are also recompiled. To ensure the derived classes are also recompiled, right-click the base class node, and then click Add-Ins > Compile forward. The alternative of clicking Build > Compile (or pressing the F7 key) is sometimes insufficient for a base class change.

A class can implement zero to many interfaces.

An X++ table implicitly inherits from the `Common` table, and from the `xRecord` class.

### Keyword Comparisons

The following table lists the OOP-related keywords in X++ and C#.

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>X++</th>
<th>C#</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract</td>
<td></td>
<td></td>
<td>No difference.</td>
</tr>
<tr>
<td>class</td>
<td>The modifiers <code>public</code> and <code>private</code> are ignored on class declarations. There is no concept of a namespace grouping of classes. There are no dots (.) in any class names.</td>
<td>The modifiers <code>public</code> and <code>private</code> can be used to modify class declarations. C# also has the keyword <code>internal</code>, which relates to how classes are grouped together in assembly files.</td>
<td>There is no concept of a <code>protected</code> class, only <code>protected</code> members of a class.</td>
</tr>
<tr>
<td>KEYWORD</td>
<td>X++</td>
<td>C#</td>
<td>COMMENTS</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>extends</td>
<td>A class declaration can inherit from another class by using the <em>extends</em> keyword.</td>
<td>A colon (:) is used where the keywords <em>extends</em> and <em>implements</em> are used in X++.</td>
<td></td>
</tr>
<tr>
<td>final</td>
<td>A <em>final</em> method cannot be overridden in a derived class. A <em>final</em> class cannot be extended.</td>
<td>The keyword <em>sealed</em> on a class means the same thing that <em>final</em> means on an X++ class.</td>
<td></td>
</tr>
<tr>
<td>implements</td>
<td>A class declaration can implement an <em>interface</em> by using the <em>implements</em> keyword.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>interface</td>
<td>An <em>interface</em> can specify methods that the class must implement.</td>
<td>An <em>interface</em> can specify methods that the class must implement.</td>
<td></td>
</tr>
<tr>
<td>new</td>
<td>The <em>new</em> keyword is used to allocate a new instance of a class. Then the constructor is automatically called. Each class has exactly one constructor, and the constructor is named <em>new</em>. You can decide what parameters the constructor should input.</td>
<td>The <em>new</em> keyword is used to create a new instance of a class. Then the constructor is automatically called. Constructor methods themselves are not named <em>new</em>, they have the same name as the class. <em>Note:</em> The <em>new</em> keyword can also be used on a method, to modify the way in which the method overrides the same method in the base class.</td>
<td>Both X++ and C# assume a default constructor for classes that have no constructor explicitly written in their code.</td>
</tr>
<tr>
<td>null</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>private and protected</td>
<td>The <em>private</em> and <em>protected</em> keywords can be used to modify the declaration of a class member.</td>
<td>The <em>private</em> and <em>protected</em> keywords can be used to modify the declaration of a class member.</td>
<td></td>
</tr>
<tr>
<td>public</td>
<td>A method that is not modified with <em>public</em>, <em>protected</em>, or <em>private</em> has the default access level of <em>public</em>.</td>
<td>A method that is not modified with <em>public</em>, <em>protected</em>, or <em>private</em> has the default access level of <em>private</em>.</td>
<td></td>
</tr>
<tr>
<td>static</td>
<td>A method can be <em>static</em>, but a field cannot.</td>
<td>Both methods and fields can be <em>static</em>.</td>
<td></td>
</tr>
<tr>
<td>KEYWORD</td>
<td>X++</td>
<td>C#</td>
<td>COMMENTS</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>super</td>
<td>The <code>super</code> keyword is used in a derived class to access the same method on its base class. void method2() { // Call method2 method // on the base class. super(); }</td>
<td>The <code>base</code> keyword is used in a derived class to access various methods in its base class. void method2() { // Call methods on // the base class. base.method2(); base.method3(); }</td>
<td>In C#, there is special syntax for using <code>base</code> to call the base constructor.</td>
</tr>
<tr>
<td>this</td>
<td>For a call from one instance method to another on the same object, a qualifier for the called method is required. The keyword <code>this</code> is available as a qualifier for the current object.</td>
<td>For a call from one instance method to another on the same object, a qualifier for the called method is not required. However, the <code>this</code> keyword is available as a qualifier for the current object. In practice, the keyword <code>this</code> can be helpful by displaying IntelliSense information.</td>
<td></td>
</tr>
<tr>
<td>finalize</td>
<td>The <code>Object</code> class contains the <code>finalize</code> method. The <code>finalize</code> method is not <code>final</code>, and it can be overridden. The <code>finalize</code> method appears to resemble the <code>System.Object.Finalize</code> method in C#, but in X++ the <code>finalize</code> method has no special meaning of any kind. An object is automatically removed from memory when the last reference to the object stops referencing the object. For example, this can happen when the last reference goes out of scope or is assigned another object to reference.</td>
<td>The methods <code>Finalize</code> and <code>Dispose</code> are common on some types of classes. The garbage collector calls the <code>Finalize</code> and <code>Dispose</code> methods when it destroys and object.</td>
<td>In C#, the <code>System.GC.Collect</code> method in the .NET Framework can be called to start the garbage collector. There is no similar function in X++ because X++ uses a deterministic garbage collector.</td>
</tr>
<tr>
<td>main</td>
<td>Classes that are invoked from a menu have their <code>main</code> method called by the system.</td>
<td>Classes that are invoked from a command line console have their <code>Main</code> method called by the system.</td>
<td></td>
</tr>
</tbody>
</table>

**Comparison: Classes**

When you use C# in the .NET Framework, classes are grouped into namespaces. Each namespace focuses on a functional area such as file operations or reflection. However, when you use the classes in X++, there are no visible groupings like a namespace.
**Comparison: Classes about Reflection**

In X++ the `TreeNode` class provides access to the Application Object Tree (AOT). The `TreeNode` class is the center of reflection functionality in X++. The `TreeNode` class and its methods can be compared to the `System.Reflection` namespace in the .NET Framework that C# uses.

The following table lists several classes that are available to you when you write C# code. These are .NET Framework classes. For this table, all C# classes are in the `System.Reflection` namespace unless otherwise specified. Each row shows the corresponding class, or class member, that is available to you when your write X++ code.

<table>
<thead>
<tr>
<th>X++</th>
<th>C#</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TreeNode</td>
<td>System .Assembly</td>
<td>Assembly is the first class to use when a C# program must gather reflection information. Static methods on the X++ class <code>TreeNode</code> are the starting point for reflection in X++.</td>
</tr>
<tr>
<td>TreeNode</td>
<td>System .Type</td>
<td>Instance methods on <code>TreeNode</code> correspond to instance methods on <code>System.Type</code></td>
</tr>
<tr>
<td>TreeNode .AOTgetSource</td>
<td>MethodInfo</td>
<td>The <code>AOTgetSource</code> method returns several pieces of information together in one string. This includes the X++ source code in the method. In contrast, <code>MethodInfo</code> has a separate member for each piece of information.</td>
</tr>
<tr>
<td>TreeNode .AOTfirstChild</td>
<td>MethodInfo[] (an array)</td>
<td>In C#, the <code>GetMethods</code> method on <code>System.Type</code> returns an array of <code>MethodInfo</code> objects. You can loop through the array by the common technique of incrementing an indexer. In contrast, the X++ model is to navigate the tree control of the AOT. The <code>TreeNode</code> methods of <code>AOTfirstChild</code> and <code>AOTnextSibling</code> accomplish the navigation. As an equivalent alternative, the X++ <code>AOTIterator</code> class is designed to navigate the tree control of the AOT. A class node is the parent over several method nodes. The <code>AOTIterator</code> steps through child nodes, returning each as another <code>TreeNode</code> instance. Additional resources the <code>TreeNode</code> methods that are named <code>AOTparent</code> and <code>AOTprevious</code>.</td>
</tr>
<tr>
<td>TreeNode .AOTgetProperties</td>
<td>PropertyInfo</td>
<td>In X++, the <code>AOTgetProperties</code> method returns a long string that contains name-value pairs for all the properties of the <code>TreeNode</code>. The <code>AOTname</code> method returns a string that contains only the value for the name property.</td>
</tr>
</tbody>
</table>
TreeNode .AOTsave
TreeNode .AOTinsert
System .Reflection .Emit
(namespace of classes)

The `AOTsave` method applies changes from a `TreeNode` object in your X++ code to the AOT, and the changes are persisted. For a large code sample, see `TreeNode.AOTsave Method`.

### Comparison: Classes about File IO

There are several classes that perform file input and output (IO) operations. In the .NET Framework that is used in C#, the counterparts to these classes reside in the `System.IO` namespace.

The following table lists several .NET Framework classes for C# that are in the `System.IO` namespace. Each row in the table shows the X++ class or method that best corresponds to the .NET Framework class.

<table>
<thead>
<tr>
<th>X++</th>
<th>C#</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinaryIo</td>
<td>FileStream, BinaryReader, BinaryWriter</td>
<td>X++ classes such as <code>BinaryIo</code> that extend from the abstract class <code>Io</code> serve as a stream, and they also serve as a reader and writer for that stream. In C#, the stream is a separate class from the class that has more specific read and write methods.</td>
</tr>
<tr>
<td>TextBuffer</td>
<td>MemoryStream</td>
<td>These classes contain an in-memory buffer, and some of the methods treat the buffer as if it were a file on the hard disk.</td>
</tr>
<tr>
<td>WINAPI::createDirectory, WINAPI::folderExists, WINAPI::removeDirectory</td>
<td>Directory, DirectoryInfo, Path</td>
<td>X++ can use static methods in the <code>WINAPI</code> class for many basic operating system functions that involve directories.</td>
</tr>
<tr>
<td>WINAPI::getDriveType</td>
<td>DriveInfo, DriveType</td>
<td>These classes and methods are used to obtain drive related information.</td>
</tr>
<tr>
<td>WINAPI::copyFile, WINAPI::createFile, WINAPI::deleteFile, WINAPI::fileExists</td>
<td>File, FileAttributes, FileInfo</td>
<td>X++ can use static methods in the <code>WINAPI</code> class for many basic operating system functions that involve files.</td>
</tr>
<tr>
<td>CommaIo, Comma7Io</td>
<td>(No corresponding class.)</td>
<td>These X++ classes can generate files that Microsoft Excel can import. In X++ an EPPlus library reference is available for additional interaction with Excel.</td>
</tr>
<tr>
<td>AsciiIo, TextIo</td>
<td>FileStream, TextReader, StreamWriter</td>
<td>These classes use different code pages.</td>
</tr>
<tr>
<td>Io</td>
<td>Stream, StreamReader, StreamWriter, FileStream</td>
<td>These are often used as base classes that other classes extend.</td>
</tr>
</tbody>
</table>
The namespace `System.Security.Permissions` includes the following classes:

- `CodeAccessSecurityAttribute`
- `FileIOPermissionAttribute`
- `FileIOPermission`
- `FileIOPermissionAccess`

The concepts and methods of `assert`, `demand`, and `revertAssert` apply to both languages. However, the `deny` and `revertDeny` methods that are available in C# are not available in X++.

### X++, ANSI SQL Comparison: SQL Select

In X++, the SQL `select` statement syntax differs from the American National Standards Institute (ANSI) specification.

#### Single Table Select

The following table lists differences between the select statements of X++ SQL and ANSI SQL.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>X++ SQL</th>
<th>ANSI SQL</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table name on the <code>from</code> clause.</td>
<td>The <code>from</code> clause lists a record buffer instance that is declared from a table, such as from the <code>CustTable</code> table.</td>
<td>The <code>from</code> clause lists a table name, not the name of a buffer.</td>
<td>The record buffer has all the methods that the <code>xRecord</code> class has in X++.</td>
</tr>
<tr>
<td>Syntax sequence of the <code>order by</code> versus <code>where</code> clauses.</td>
<td>The order by clause must appear before the <code>where</code> clause. The order by clause must appear after the <code>from</code> or <code>join</code> clause. The group by clause must follow the same syntax positioning rules that the order by follows.</td>
<td>The order by clause must appear after the <code>where</code> clause. The <code>where</code> clause must appear after the <code>from</code> or <code>join</code> clause.</td>
<td>In both X++ and ANSI SQL, the <code>from</code> and <code>join</code> clauses must appear before the <code>order by</code> and <code>where</code> clauses.</td>
</tr>
<tr>
<td>Condition negation.</td>
<td>The exclamation mark (<code>!</code>) is used for negation.</td>
<td>The <code>not</code> keyword is used for negation.</td>
<td>X++ does not support the syntax <code>!like</code>. Instead, you must apply the <code>!</code> operator to a clause.</td>
</tr>
<tr>
<td>Wildcard characters for the <code>like</code> operator.</td>
<td>0 to many – Asterisk (<code>*</code>) Exactly 1 – Question mark (<code>?</code>)</td>
<td>0 to many – Percent sign (<code>%</code>) Exactly 1 – Underbar (<code>_</code>)</td>
<td></td>
</tr>
<tr>
<td>Logical operators in the <code>where</code> clause.</td>
<td>And – <code>&amp;&amp;</code> Or – `</td>
<td></td>
<td>`</td>
</tr>
</tbody>
</table>

#### Code Example

The following code example illustrates features in the previous table.
static void OByWhere452Job(Args _args)
{
    // Declare the table buffer variable.
    CustTable tCustTable;
    
    SELECT * from tCustTable
    order by tCustTable.AccountNum desc
    where (!(tCustTable.Name like '*i*i*') && tCustTable.Name like 'T?e ')
    {
        info(tCustTable.AccountNum + " , " + tCustTable.Name);
    }
}

//*** InfoLog output
Message (04:02:29 pm)
4010 , The Lamp Shop
4008 , The Warehouse
4001 , The Bulb
***/

**X++ SQL Keywords**

The following X++ SQL keywords are among those that are not part of ANSI SQL:

- crosscompany
- firstonly100
- forceliterals
- forcenestedloop
- forceplaceholders
- forceselectorder
- validtimestate

**Join Clause**

The following table lists differences about the `join` keyword of X++ SQL and ANSI SQL.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>X++ SQL</th>
<th>ANSI SQL</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns list</td>
<td>The columns in the columns list must all come from the table listed in the <code>from</code> clause, and not from any table in a <code>join</code> clause. Columns in the list cannot be qualified by their table name.</td>
<td>The columns in the columns list can come from any table in the <code>from</code> or <code>join</code> clauses. It helps others to maintain your code when you qualify the columns in the list with their table name.</td>
<td>For more information, see Select Statements on Fields.</td>
</tr>
<tr>
<td>Join clause syntax</td>
<td>The <code>join</code> clause follows the <code>where</code> clause.</td>
<td>The <code>join</code> clause follows a table in the <code>from</code> clause.</td>
<td>In the X++ code example, the <code>join</code> criteria is an equality of <code>$SalesPoolId</code> values.</td>
</tr>
<tr>
<td>Inner keyword</td>
<td>The default <code>join</code> mode is inner join. There is no <code>inner</code> keyword.</td>
<td>The default <code>join</code> mode is inner join. The <code>inner</code> keyword is available to make the code explicit.</td>
<td>The <code>outer</code> keyword exists in both X++ SQL and ANSI SQL.</td>
</tr>
<tr>
<td>Left and right keywords</td>
<td>The <code>left</code> and <code>right</code> keywords are not available. All joins are left.</td>
<td>The <code>left</code> and <code>right</code> keywords are available to modify the <code>join</code> keyword.</td>
<td>No comments.</td>
</tr>
</tbody>
</table>
Equality operator.
The double equal sign operator ("==") is used to test for the equality of two values.

The single equal sign operator ("=") is used to test for the equality of two values.

No comments.

Code Example
The following code example illustrates the `join` syntax in X++ SQL.

```java
static void OByWhere453Job(Args _args)
{
    // Declare table buffer variables.
    CustTable tCustTable;
    SalesPool tSalesPool;

    while
        SELECT
            AccountNum,
            Name
        from tCustTable
        order by tCustTable.AccountNum desc
        where (tCustTable.Name like 'The *')
        join tSalesPool
            where tCustTable.SalesPoolId == tSalesPool.SalesPoolId
    {
        info(tCustTable.AccountNum + " , " + tCustTable.Name);
    }
}
```

Aggregate Fields
The following table lists some differences in how aggregate fields in the `select` column list are referenced between X++ SQL and ANSI SQL. Aggregate fields are those that are derived by functions such as `sum` or `avg`.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>X++ SQL</th>
<th>ANSI SQL</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregate field name alias.</td>
<td>The aggregate value is in the field that was aggregated.</td>
<td>You can use the <code>as</code> keyword to tag an aggregate field with a name alias. The alias can be referenced in subsequent code.</td>
<td>For more information, see Aggregate Functions: Differences Between X++ and SQL</td>
</tr>
</tbody>
</table>

Code Example
In the following code example, the call to the info method illustrates the way to reference aggregate fields (see `tPurchLine.QtyOrdered`).
static void Null673Job(Args _args)
{
    PurchLine tPurchLine;
    while
        select
            // This aggregate field cannot be assigned an alias name.
            sum(QtyOrdered)
        from tPurchLine
    {
        info(
            // QtyOrdered is used to reference the sum.
            "QtyOrdered: " + num2str(tPurchLine.QtyOrdered,
            3, // Minimum number of output characters.
            2, // Required number of decimal places in the output.
            1, // ' .' Separator to mark the start of the decimal places.
            2 // ',' The thousands separator.
            ));
        info("End.");
    }
/**
Message (12:23:08 pm)
QtyOrdered:  261,550.00
End.
***/

Other Differences

The following table lists other differences of the select statement between the X++ SQL and ANSI SQL.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>X++ SQL</th>
<th>ANSI SQL</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>The having keyword.</td>
<td>There is no having keyword.</td>
<td>The having keyword enables you to specify filter criteria for rows that are generated by the group by clause.</td>
<td>No comments.</td>
</tr>
<tr>
<td>Null results.</td>
<td>In a while select statement, if the where clause filters out all rows, no special count row is returned to report that.</td>
<td>In a select, if the where clause filters out all rows, a special count row is returned. The count value is 0.</td>
<td>No comments.</td>
</tr>
<tr>
<td>Cursors for navigating returned rows.</td>
<td>The while select statement provides cursor functionality. The alternative is to use the next keyword.</td>
<td>You can declare a cursor for looping through the rows that are returned from a select statement.</td>
<td></td>
</tr>
<tr>
<td>From clause.</td>
<td>The from keyword is optional when no columns are listed and only one table is referenced. The following two syntax options are equivalent:</td>
<td>A select statement cannot read from a table unless the from clause is used.</td>
<td>In X++ SQL, the simple select statement fills the table buffer variable with the first row that was returned. This is illustrated by the following code fragment:</td>
</tr>
</tbody>
</table>
|              | select \\
|              | tCustTable;
|              | select tCustTable; | select \\
|              | tCustTable;
|              | info(tCustTable.Name); |
This topic contains the syntax reference for X++.

### X++ Keywords

The X++ keywords shown in the following table are reserved. These keywords cannot be used for any other purpose.

<table>
<thead>
<tr>
<th>RESERVED WORD</th>
<th>DESCRIPTION</th>
<th>MORE INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>!</code></td>
<td>Not.</td>
<td>Relational Operators</td>
</tr>
<tr>
<td><code>!=</code></td>
<td>Inequality operator (not equal to).</td>
<td>Relational Operators</td>
</tr>
<tr>
<td><code>#</code></td>
<td>Prefix on macro names.</td>
<td>How to: Use <code>#define</code> and <code>#if</code> to Test a Macro</td>
</tr>
<tr>
<td><code>&amp;</code></td>
<td>Binary AND.</td>
<td>Arithmetic Operators</td>
</tr>
<tr>
<td><code>&amp;&amp;</code></td>
<td>Logical AND.</td>
<td>Relational Operators</td>
</tr>
<tr>
<td><code>(</code></td>
<td>Function call operator, which indicates the beginning of the function call.</td>
<td></td>
</tr>
<tr>
<td><code>)</code></td>
<td>Function call operator, which indicates the end of the function call.</td>
<td></td>
</tr>
<tr>
<td><code>*</code></td>
<td>Multiply. The asterisk (*) is also used in X++ SQL. One use is to signify all fields from the tables on a <code>select</code> statement. Another use is as a wildcard with the <code>like</code> operator, to signify 0 to many characters of any kind. The <code>like</code> operator also uses the ? character.</td>
<td>Arithmetic Operators</td>
</tr>
<tr>
<td><code>^</code></td>
<td>Binary XOR.</td>
<td>Arithmetic Operators</td>
</tr>
<tr>
<td>`</td>
<td>`</td>
<td>Binary OR.</td>
</tr>
<tr>
<td>`</td>
<td></td>
<td>`</td>
</tr>
<tr>
<td><code>~</code></td>
<td>Not.</td>
<td>Arithmetic Operators</td>
</tr>
<tr>
<td><code>+</code></td>
<td>Plus.</td>
<td>Arithmetic Operators</td>
</tr>
<tr>
<td><code>++</code></td>
<td>Increment.</td>
<td>Assignment Operators</td>
</tr>
<tr>
<td><code>+=</code></td>
<td>Additive assignment.</td>
<td>Assignment Operators</td>
</tr>
<tr>
<td>RESERVED WORD</td>
<td>DESCRIPTION</td>
<td>MORE INFORMATION</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>------------------</td>
</tr>
<tr>
<td><code>,</code></td>
<td>Comma operator. Expressions separated by commas are evaluated left-to-right.</td>
<td></td>
</tr>
<tr>
<td><code>-</code></td>
<td>Minus.</td>
<td>Arithmetic Operators</td>
</tr>
<tr>
<td><code>--</code></td>
<td>Decrement operator.</td>
<td>Assignment Operators</td>
</tr>
<tr>
<td><code>-=</code></td>
<td>Subtractive assignment.</td>
<td>Assignment Operators</td>
</tr>
<tr>
<td><code>.</code></td>
<td>Class member access operator, for example, <code>FormRun.run</code> accesses the <code>run</code> method of an object of the class type <code>FormRun</code>.</td>
<td></td>
</tr>
<tr>
<td><code>/</code></td>
<td>Divide.</td>
<td>Arithmetic Operators</td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>Less than.</td>
<td>Relational Operators</td>
</tr>
<tr>
<td><code>&lt;&lt;</code></td>
<td>Left shift.</td>
<td>Arithmetic Operators</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>Less than or equal.</td>
<td>Arithmetic Operators</td>
</tr>
<tr>
<td><code>=</code></td>
<td>Assignment operator. The argument to the left of <code>=</code> is set to the value of the argument to the right.</td>
<td>Assignment Operators</td>
</tr>
</tbody>
</table>

</strong> Escape in strings. Escapes extra quotation marks, and certain letters such as `\t` for tab.

@ Escape of keywords. For example, `str @abstract;` would fail to compile without the `@` sign. Also affects literal strings, by negating the effect of the `\` escape character, and by enabling the string to span more than one line in the source code. The new line is represented by one character of hexadecimal 0x0A, which is commonly called a line feed. No carriage return character of hexadecimal 0x0D is included, as in 0x0D0A.

Field declaration or label specifier. The colon (:) character is also used on the `switch` statement.

:: Used to call static (class) methods: `ClassName::methodName`.

; Terminates statements. Used in `for` loops or as a separator of statements.
<table>
<thead>
<tr>
<th>RESERVED WORD</th>
<th>DESCRIPTION</th>
<th>MORE INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>Returns true if both expressions are equal.</td>
<td>Relational Operators</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than.</td>
<td>Relational Operators</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal.</td>
<td>Relational Operators</td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>Right shift.</td>
<td>Arithmetic Operators</td>
</tr>
<tr>
<td>?</td>
<td>Ternary operator. The question mark (?) character is also used by the <code>like</code> operator to signify exactly one character of any kind. The <code>like</code> operator also uses the character.</td>
<td>Ternary Operator (?)</td>
</tr>
<tr>
<td>[</td>
<td>Array declarator, open. Must be used with <code>]</code>.</td>
<td></td>
</tr>
<tr>
<td>]</td>
<td>Array declarator, close. Must be used with <code>[</code>.</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td>Indicates the beginning of a number of statements. The last of these statements must be followed by a <code>}</code>.</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td>Indicates the end of a number of statements. A <code>(</code> must appear before the first of these statements.</td>
<td></td>
</tr>
<tr>
<td><strong>abstract</strong></td>
<td>Class and method modifier. An <strong>abstract</strong> class cannot be constructed with the <strong>new</strong> keyword. An <strong>abstract</strong> method cannot be called. A table can also be modified as abstract by setting its Abstract property to Yes in the AOT, or by using the <strong>DictTable</strong> class. The Abstract property defaults to No, and it cannot be set unless the table is extended by another table. Each row in an abstract table must have a dependent row in a derived table. This means that each row in an abstract table has a value greater than 0 (zero) in its InstanceRelationType property field. There are no other effects from marking a table as abstract. Informally, programmers often use the term concrete to describe a class that is non-<strong>abstract</strong>.</td>
<td>Method Modifiers Table Inheritance Overview</td>
</tr>
<tr>
<td><strong>anytype</strong></td>
<td>The method can return any data type.</td>
<td>Anytype</td>
</tr>
<tr>
<td>RESERVED WORD</td>
<td>DESCRIPTION</td>
<td>MORE INFORMATION</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>------------------</td>
</tr>
<tr>
<td>as</td>
<td>Needed when you assign a base class variable to a derived class variable. For example, given a <code>Derived</code> class that extends a <code>Base</code> class, the statement <code>myDerived = myBase as Derived;</code> avoids a compiler error by using the <code>as</code> keyword. This keyword also applies when you assign a base table variable to a derived table variable.</td>
<td>Expression Operators: Is and As for Inheritance</td>
</tr>
<tr>
<td>asc</td>
<td>An option on the <code>order by</code> or <code>group by</code> clause in a <code>select</code> statement. The sorting is ascending.</td>
<td>Select Statement Syntax</td>
</tr>
<tr>
<td>at</td>
<td>Specifies the position of a print window.</td>
<td>Print Statements</td>
</tr>
<tr>
<td>avg</td>
<td>Returns the average of the fields from the rows specified by the <code>group by</code> clause in a <code>select</code> statement.</td>
<td>Select Statement Syntax</td>
</tr>
<tr>
<td>break</td>
<td>Immediate exit from code block.</td>
<td>Break Statements</td>
</tr>
<tr>
<td>breakpoint</td>
<td>Represents a breakpoint that is set for debugging purposes. To set a breakpoint in your code, write: <code>breakpoint;</code></td>
<td></td>
</tr>
<tr>
<td>by</td>
<td>Part of a reserved term, such as group by and order by.</td>
<td></td>
</tr>
<tr>
<td>byref</td>
<td>Specifies that the parameter being passed to the called method is being passed by reference (address), instead of by value. <code>Byref</code> is used in X++ when calling a .NET method that takes a parameter by reference (such as with the C# keywords <code>out</code> or <code>ref</code>).</td>
<td>How to: Use the byref Keyword for CLR Interop.</td>
</tr>
<tr>
<td>case</td>
<td>Selection within a <code>switch</code> statement.</td>
<td>Switch Statements</td>
</tr>
<tr>
<td>catch</td>
<td>Used in exception handling.</td>
<td>Exception Handling with try and catch Keywords</td>
</tr>
<tr>
<td>changeCompany</td>
<td>Changes database settings to another company.</td>
<td>Change Company Design Pattern</td>
</tr>
<tr>
<td>class</td>
<td>Declares a class.</td>
<td>Classes in X++</td>
</tr>
<tr>
<td>client</td>
<td>Method modifier.</td>
<td>Method Modifiers</td>
</tr>
<tr>
<td>container</td>
<td>Specifies a variable of type <code>container</code>.</td>
<td>Containers</td>
</tr>
<tr>
<td>RESERVED WORD</td>
<td>DESCRIPTION</td>
<td>MORE INFORMATION</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>------------------</td>
</tr>
<tr>
<td>continue</td>
<td>Forces the next iteration of a loop.</td>
<td>Continue Statements</td>
</tr>
<tr>
<td>count</td>
<td>Returns the number of records from the rows specified by the ( \text{group by} ) clause in a <code>select</code> statement.</td>
<td>Select Statement Syntax</td>
</tr>
<tr>
<td>crossCompany</td>
<td>Causes a <code>select</code> statement to return data for all companies that the user is authorized to read from.</td>
<td>Cross-Company X++ Code Basics</td>
</tr>
<tr>
<td>date</td>
<td>Specifies a variable of type <code>date</code>.</td>
<td>Dates</td>
</tr>
<tr>
<td>default</td>
<td>Default case within <code>switch</code> statements.</td>
<td>Switch Statements</td>
</tr>
</tbody>
</table>
| delegate      | A class member that is able to store multiple references to methods in other classes, and to call all those methods when prompted to do so. A delegate can store references to various kinds of methods including the following:  
  • static methods on X++ classes  
  • instance methods on X++ classes  
  • methods on .NET Framework classes | Event Terminology and Keywords X++, C# Comparison: Event |
<p>| delete_from   | Allows you to delete multiple records from the database at the same time. | delete_from |
| desc          | An option on the <code>order by</code> or <code>group by</code> clause in a <code>select</code> statement. The sorting is descending. | Select Statement Syntax |
| display       | Method modifier. | Method Modifiers |
| div           | Integer division. | Arithmetic Operators |
| do            | Beginning of a <code>do...while</code> loop. | Do...while Loops |
| edit          | Method modifier. | Method Modifiers |
| else          | Conditional execution (<code>if...else</code>). | <code>if</code> and <code>if ... else</code> Statements |
| eventHandler  | Must be used each time you either add or delete a method reference from a delegate by using the <code>+=</code> or <code>-=</code> operator. For example: <code>myDelegate += eventHandler(OtherClass::myStaticMethod);</code> | Event Terminology and Keywords X++, C# Comparison: Event |</p>
<table>
<thead>
<tr>
<th>RESERVED WORD</th>
<th>DESCRIPTION</th>
<th>MORE INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>exists</td>
<td>Used with <code>join</code> clauses in <code>select</code> statements.</td>
<td>Select Statement Syntax</td>
</tr>
<tr>
<td>extends</td>
<td>A class or interface declaration clause. If your class does not explicitly extend another class, your class is considered to extend the <code>Object</code> class (as if you had written &quot;extends Object&quot;).</td>
<td>Creating a Subclass</td>
</tr>
<tr>
<td>false</td>
<td>Boolean literal.</td>
<td>Booleans</td>
</tr>
<tr>
<td>final</td>
<td>Class and method modifier.</td>
<td>Method Modifiers</td>
</tr>
<tr>
<td>firstFast</td>
<td>Used in <code>select</code> statements to speed up the fetch for the first row.</td>
<td>Select Statement Syntax</td>
</tr>
<tr>
<td>firstOnly</td>
<td>Used in <code>select</code> statements to fetch only the first record. The <code>firstOnly</code> keyword does not guarantee that a maximum of one record is retrieved by an X++ SQL <code>select</code> statement. If the AOS can use the <code>EntireTable</code> cache to satisfy the data demands of the <code>select</code> statement, the <code>firstOnly</code> keyword is ignored.</td>
<td>Select Statement Syntax Set-based Caching</td>
</tr>
<tr>
<td>firstOnly10</td>
<td>Same as <code>firstOnly</code>, except returns 10 rows instead of one.</td>
<td></td>
</tr>
<tr>
<td>firstOnly100</td>
<td>Same as <code>firstOnly</code>, except returns 100 rows instead of one.</td>
<td></td>
</tr>
<tr>
<td>firstOnly1000</td>
<td>Same as <code>firstOnly</code>, except returns 1000 rows instead of one.</td>
<td></td>
</tr>
<tr>
<td>flush</td>
<td>Clears an entire table cache. Here is the syntax for the <code>flush</code> statement: <code>YourTable ytBuffer; flush ytBuffer;</code></td>
<td>Set-based Caching</td>
</tr>
<tr>
<td>for</td>
<td>For loop iteration.</td>
<td>For Loops</td>
</tr>
<tr>
<td>forceLiterals</td>
<td>Used in <code>select</code> statements to reveal actual values that are used in <code>where</code> clauses to the Microsoft SQL Server database at the time of optimization.</td>
<td>Select Statement Syntax</td>
</tr>
<tr>
<td>forceNestedLoop</td>
<td>Forces the SQL Server database to use a nested-loop algorithm to process a particular SQL statement containing a <code>join</code>.</td>
<td>Select Statement Syntax</td>
</tr>
<tr>
<td>RESERVED WORD</td>
<td>DESCRIPTION</td>
<td>MORE INFORMATION</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>forcePlaceholders</td>
<td>Used in <code>select</code> statements to instruct the kernel not to reveal the actual values used in <code>where</code> clauses to the Microsoft SQL Server database at the time of optimization.</td>
<td>Select Statement Syntax</td>
</tr>
<tr>
<td>forceSelectOrder</td>
<td>Forces the SQL Server database to access the tables in a join in the specified order.</td>
<td>Select Statement Syntax</td>
</tr>
<tr>
<td>forUpdate</td>
<td>Selects records exclusively for update. The operation to be performed on the records that are fetched is an update. Depending on the underlying database, the records may be locked for other users.</td>
<td>Select Statement Syntax</td>
</tr>
<tr>
<td>from</td>
<td>Part of a <code>select</code> statement. The <code>from</code> clause specifies the table in which the columns exists.</td>
<td>Select Statement Syntax</td>
</tr>
<tr>
<td>group</td>
<td>Part of the <code>group by</code> clause in a <code>select</code> statement.</td>
<td>Select Statement Syntax</td>
</tr>
<tr>
<td>if</td>
<td>Conditional execution.</td>
<td>if and if ... else Statements</td>
</tr>
<tr>
<td>implements</td>
<td>Implements an interface.</td>
<td>Interfaces Overview</td>
</tr>
<tr>
<td>insert_recordset</td>
<td>Copies data from one or more tables into one resulting destination table on a single server trip.</td>
<td>insert_recordset</td>
</tr>
<tr>
<td>int</td>
<td>Specifies a variable of type <code>integer</code> (32-bit).</td>
<td>Integers</td>
</tr>
<tr>
<td>int64</td>
<td>Specifies a variable of type <code>integer</code> (64-bit).</td>
<td>Integers</td>
</tr>
<tr>
<td>interface</td>
<td>Interface declaration.</td>
<td>Interfaces Overview</td>
</tr>
<tr>
<td>is</td>
<td>Asks whether the object referenced by a class variable either inherits from the given class or is of the given class. For example, given a <code>Derived</code> class that <code>extends</code> a <code>Base</code> class, the expression <code>(myDerived is Base)</code> returns <code>true</code>. This keyword applies to class inheritance and table inheritance.</td>
<td>Expression Operators: Is and As for Inheritance</td>
</tr>
<tr>
<td>join</td>
<td>Tables are joined on columns common to both tables. You can generate a single result set based on multiple tables through the use of <code>joins</code>.</td>
<td>Select Statement Syntax</td>
</tr>
<tr>
<td>RESERVED WORD</td>
<td>DESCRIPTION</td>
<td>MORE INFORMATION</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| like          | Tests for matches by pattern, with wildcard symbols * and ?. The string on the right side of the `like` operator must use four backslash characters to represent one backslash. Examples follow:  
  • `("&quot; like "&lt;em&gt;")`  
    //Resolves to false.  
  • `("&quot; like "\+")`  
    //Resolves to true. | Relational Operators |
| maxof         | Returns the maximum of the fields from the rows specified by the `group by` clause. | Select Statement Syntax |
| minof         | Returns the minimum of the fields from the rows specified by the `group by` clause. | Select Statement Syntax |
| mod           | Returns the integer remainder of the left expression1 divided by the right expression2. Informally this is sometimes called the modulo operator.  
  
  
  
  
  
<p>| ((12 mod 7) == 5) is true. | |
| new           | Operator. Creates an instance of an anonymous class that is assignment-compatible with the named class/interface reference variables, or allocates memory for an array. | |
| next          | Fetches the next record in a table. | |
| noFetch       | Indicates that no records are to be fetched at present. | Select Statement Syntax |
| notExists     | Used with <code>join</code> clauses in <code>select</code> statements. | Select Statement Syntax |
| null          | Symbolic constant. | |
| optimisticLock| Forces a statement to run with optimistic concurrency control, even if a different value is set on the table. | Select Statement Syntax |
| order         | Part of the <code>order by</code> clause in a <code>select</code> statement. | Select Statement Syntax |
| outer         | outer join. | Select Statement Syntax |
| pause         | Halts the execution of a job. The user is asked to state whether execution should continue. | Select Statements |</p>
<table>
<thead>
<tr>
<th>RESERVED WORD</th>
<th>DESCRIPTION</th>
<th>MORE INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>pessimisticLock</td>
<td>Forces a statement to run with pessimistic concurrency control, even if a different value is set on the table.</td>
<td>Select Statement Syntax</td>
</tr>
<tr>
<td>print</td>
<td>Allows you to display output on the screen.</td>
<td>Print Statements</td>
</tr>
<tr>
<td>private</td>
<td>Method access modifier.</td>
<td>Method Access Control</td>
</tr>
<tr>
<td>protected</td>
<td>Method access modifier.</td>
<td>Method Access Control</td>
</tr>
<tr>
<td>public</td>
<td>Method access modifier.</td>
<td>Method Access Control</td>
</tr>
<tr>
<td>real</td>
<td>Specifies a variable of type real.</td>
<td>Reals</td>
</tr>
<tr>
<td>repeatableRead</td>
<td>Specifies that no other transactions can modify data that has been read by logic inside the current transaction, until after the current transaction completes. An explicit transaction completes at either ttsAbort or at the outermost ttsCommit. For a stand-alone select statement, the transaction duration is the duration of the select command. However, the database sometimes enforces the equivalent of repeatableRead in individual select statements even without this keyword appearing in your X++ code (depending on how the database decides to scan the tables).</td>
<td>For more information, see the documentation for the underlying relational database product.</td>
</tr>
<tr>
<td>retry</td>
<td>Used in exception handling.</td>
<td>Exception Handling with try and catch Keywords</td>
</tr>
<tr>
<td>return</td>
<td>Exits from a method.</td>
<td>Declaration of Methods</td>
</tr>
<tr>
<td>reverse</td>
<td>Records are returned in reverse order.</td>
<td>Select Statement Syntax</td>
</tr>
<tr>
<td>select</td>
<td>The select clause designates which columns or views are shown in the result set.</td>
<td>Select Statements</td>
</tr>
<tr>
<td>server</td>
<td>Method modifier.</td>
<td>Method Modifiers</td>
</tr>
<tr>
<td>setting</td>
<td>Used with the update_recordset command.</td>
<td>update_recordset</td>
</tr>
<tr>
<td>static</td>
<td>Static methods may not refer to instance variables (only to static variables); may be invoked by using the class name rather than on an instance of the class (&quot;MyClass.aStaticProcedure&quot;).</td>
<td>Method Modifiers</td>
</tr>
<tr>
<td>RESERVED WORD</td>
<td>DESCRIPTION</td>
<td>MORE INFORMATION</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>------------------</td>
</tr>
<tr>
<td>str</td>
<td>Specifies a variable of type <code>string</code>.</td>
<td>Strings</td>
</tr>
<tr>
<td>sum</td>
<td>Returns the sum of the fields from the rows specified by the <code>group by</code> clause in a <code>select</code> statement.</td>
<td>Select Statement Syntax</td>
</tr>
<tr>
<td>super</td>
<td>Calls the method that was overridden by the current method.</td>
<td>Table Methods</td>
</tr>
<tr>
<td>switch</td>
<td>Switch selection statement.</td>
<td>Switch Statements</td>
</tr>
<tr>
<td>tableLock</td>
<td>Obsolete; tableLock is no longer available.</td>
<td></td>
</tr>
<tr>
<td>this</td>
<td>A reference to the current instance of the class. Used in X++ code inside a method of the class. Used to reference <code>method</code> members of the class, but not <code>field</code> members of the class.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>public str getFullName() { // Next statement fails to compile without 'this'.</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>return this.concatenateFirstAndLastNames(); }</code></td>
<td></td>
</tr>
<tr>
<td>throw</td>
<td>Used in exception handling.</td>
<td>Exception Handling with try and catch Keywords</td>
</tr>
<tr>
<td>true</td>
<td>Boolean literal.</td>
<td>Booleans</td>
</tr>
<tr>
<td>try</td>
<td>Used in exception handling.</td>
<td>Exception Handling with try and catch Keywords</td>
</tr>
<tr>
<td>ttsAbort</td>
<td>Discards all changes in the current transaction.</td>
<td>Transaction Integrity</td>
</tr>
<tr>
<td>ttsBegin</td>
<td>Marks the beginning of a transaction.</td>
<td>Transaction Integrity</td>
</tr>
<tr>
<td>ttsCommit</td>
<td>Marks the end of a transaction.</td>
<td>Transaction Integrity</td>
</tr>
<tr>
<td>update_recordset</td>
<td>Allows the manipulation of row sets within one operation.</td>
<td>update_recordset</td>
</tr>
<tr>
<td>validTimeState</td>
<td>Filters rows that are retrieved from a valid time state table by an X++ SQL <code>select</code> statement. For example: select validTimeState(myDateEffective) * from xMyTable; ... or... select validTimeState(myDateFrom, myDateTo) * from xMyTable;</td>
<td>Effects of Valid Time State Tables on Read and Write Operations</td>
</tr>
<tr>
<td>void</td>
<td>Identifies a method that does not return a value.</td>
<td>Declaration of Methods</td>
</tr>
</tbody>
</table>
**Reserved Word** | **Description** | **More Information**
--- | --- | ---
**where** | Part of a `select` statement. The `where` clause specifies the conditions to be satisfied; that is, the rows that you want to include in the result. | Select Statement Syntax

**while** | Iteration statement. Executes a statement or block repeatedly when a test condition is true. | While Loops while select Statements

**window** | Allows you to alter the size of the output window. | Print Statements

---

**Expressions Syntax**

An expression in X++ is used in either a mathematical or logical way. Expressions are built on the data types of the language; that is, an expression returns a value of some type. This value can be used in calculations, assignments, conditional statements, and so on.

**EBNF Description of Expressions in X++**

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression</td>
<td>= Simple-expression [RelationalOperator Simple-expression ]</td>
</tr>
<tr>
<td>RelationalOperator</td>
<td>=</td>
</tr>
<tr>
<td>Simple-expression</td>
<td>= Simple-expression [ +</td>
</tr>
<tr>
<td>Term</td>
<td>= Compfactor ( Mult-operator CompFactor )</td>
</tr>
<tr>
<td>Mult-operator</td>
<td>= *</td>
</tr>
<tr>
<td>CompFactor</td>
<td>= [ ] [ +</td>
</tr>
<tr>
<td>Factor</td>
<td>= Literal</td>
</tr>
<tr>
<td>Enum</td>
<td>= EnumName :: Literal</td>
</tr>
<tr>
<td>Variable</td>
<td>= Identifier [ Expression ]</td>
</tr>
<tr>
<td>FunctionCall</td>
<td>= [ Expression</td>
</tr>
<tr>
<td>If-expression</td>
<td>= Expression ? Expression : Expression</td>
</tr>
</tbody>
</table>

Semantic restrictions apply on the preceding syntax. You cannot call any method using the :: operator. Similarly, you cannot use the this keyword without an active object; that is, if you are not within a method and so on.

**Examples**
### Example of Expression

<table>
<thead>
<tr>
<th>EXAMPLE OF EXPRESSION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>An integer literal.</td>
</tr>
<tr>
<td>NoYes::No</td>
<td>An enum-reference.</td>
</tr>
<tr>
<td>A</td>
<td>A variable-reference.</td>
</tr>
<tr>
<td>Debtor::Find(&quot;1&quot;)</td>
<td>A static method-call (returns a customer variable).</td>
</tr>
<tr>
<td>(A &gt; 3 ? true : false)</td>
<td>An if-expression that returns true or false.</td>
</tr>
<tr>
<td>(select CustTable where CustTable.Account == &quot;100&quot;).NameRef</td>
<td>A select-expression. Returns the nameref field in the customer table. This is a string.</td>
</tr>
<tr>
<td>A &gt;= B</td>
<td>A logical expression. Returns true or false.</td>
</tr>
<tr>
<td>A + B</td>
<td>An arithmetic expression. Sums A and B.</td>
</tr>
<tr>
<td>A + B / C</td>
<td>Calculates B/C, and then adds this to A.</td>
</tr>
<tr>
<td>~A + this.Value()</td>
<td>Sums binary not A and the result of the method-call Value on the object in scope (this).</td>
</tr>
<tr>
<td>Debtor::Find(&quot;1&quot;).NameRef</td>
<td>Returns the NameRef field of the found customer record.</td>
</tr>
<tr>
<td>Debtor::Find(&quot;1&quot;).Balance()</td>
<td>A method call to Balance in the customer table (Debtor::Find returns a customer). Returns the balance of the customer with account number 1.</td>
</tr>
</tbody>
</table>

### EBNF Overview

Extended Backus Naur Form (EBNF) is a metalanguage and is used in this guide to describe the language syntax. An EBNF definition consists of production rules, nonterminals, and terminals. The key terms are shown in the following table.

<table>
<thead>
<tr>
<th>KEY TERMS</th>
<th>EXAMPLE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminals</td>
<td>Work_Team</td>
<td>A terminal is one character or a string of characters that never change.</td>
</tr>
<tr>
<td>Nonterminals</td>
<td>Employee</td>
<td>A nonterminal is a description of part of a valid sentence in the language that is defined either by a production rule or a textual description. A nonterminal symbol can always be expanded to one or more terminal symbols.</td>
</tr>
<tr>
<td>Production rules</td>
<td>Employee = Developer</td>
<td>Tester This example defines a Work_Team as consisting of a Manager and one or more Employees. An Employee is defined as being a</td>
</tr>
</tbody>
</table>

**Example**

Work_Team = Manager Employee {, Employee}   Employee = Developer | Tester This example defines a Work_Team as consisting of a Manager and one or more Employees. An Employee is defined as being a
The symbols used in the example are described in the following table.

**Special Symbols in EBNF**

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Expression)</td>
<td>Parentheses hold the symbols (terminals and nonterminals) together. They can be placed anywhere on the right side of a production rule.</td>
</tr>
<tr>
<td>Expression1</td>
<td></td>
</tr>
<tr>
<td>[Expression]</td>
<td>Optional: The items between [ and ] are optional. All or none of the items in the brackets are included.</td>
</tr>
<tr>
<td>{Expression}</td>
<td>Repeat: The items between { and } are optional, but can be repeated as many times as necessary.</td>
</tr>
</tbody>
</table>

For example, if the accessories you buy for your bicycle consist of a saddle, water-bottle holders, bells, and horns, and you could have either a bell or a horn, and zero, one, or more water bottle holders, and exactly one saddle, this could be expressed as: Bicycle_Accessories = saddle [bell | horn] {water_bottle_holders} This grammar defines the following possibilities: saddle saddle bell saddle horn saddle water_bottle_holder saddle bell water_bottle_holder saddle bell water_bottle_holder water_bottle_holder And so on.

### X++ Grammar

This topic shows the formal grammar of the X++ language.

**How to Interpret the Formal BNF Grammar**

This section describes the grammar of X++ in Backus Naur Form (BNF). A small example of BNF is described here.

```
AA ::= BB CC_SYM
BB ::= JJ_SYM
::= KK_SYM
```

**AA** is the name of a production rule. An **AA** requires a **BB**, followed by a CC_SYM. A **BB** is also a production rule. Therefore, **BB** is not a terminal. **BB** must be either a JJ_SYM or a KK_SYM. Both JJ_SYM and KK_SYM are terminals because they are not the names of any other production rules. CC_SYM is also a terminal.

In the BNF for X++ grammar, most of the terminals have _SYM as the suffix of their name.

### The Formal X++ Grammar in BNF

This section contains the BNF that defines the grammar of X++.

```
CMPL_UNIT ::= RETTYPEID FUNC_HDR FUNC_HEAD BODY
::= RETTYPEID DATA_HDR CLASS_DECL
::= EXPR_HDR IF_EXPR SEMIOPT
::= RETTYPEID FUNC_HDR EVENT_DECL BODY
SEMIOPT ::= SEMICOLON_SYM
::= SEMICOLON_SYM
::= SEMICOLON_SYM
CLASS_DECL ::= CLASS_HEADER LEFTBR_SYM DCL_EVENTMAP DCL_LIST RIGHTBR_SYM
CLASS_HEADER ::= ATTRIBUTE_DEF CLASS_MODIFIERS CLASSINTERFACE STD_ID EXTENDS IMPLEMENTS
ATTRIBUTE_DEF ::= LEFT_BRKT_SYM ATTRIBUTE_INIT ATTRIBUTE_LIST RETTYPEID RGHT_BRKT_SYM
::=
ATTRIBUTE_INIT ::= 
```
LOCAL_BODY ::= LEFTBR_SYM DCL_LIST SEMIOPT STMTLIST RETTYPEID RIGHTBR_SYM
DCL_LIST ::= DCL_LIST2
::=
 DCL_LIST2 ::= DCL_STMT
 ::= DCL_LIST2 DCL_STMT
DCL_FUNC_LIST ::= DCL_FUNC_LIST2
::=
 DCL_FUNC_LIST2 ::= DCL_STMT
 ::= DCL_FUNC_LIST2 FUNCTION_DEF
DCL_STMT ::= DCL_INIT_LIST RETTYPEID SEMICOLON_SYM
DCL_INIT_LIST ::= DCL_INIT
 ::= DCL_CLIST ASG_CLAUSE
DCL_CLIST ::= DCL_INIT_LIST LIST_SEP_SYM STD_ID ARR_DCL_IDX
DCL_INIT ::= DECL ASG_CLAUSE
DECL ::= DECL_TYPE STD_ID ARR_DCL_IDX
DECL_TYPE ::= STR_TYPE_SYM STR_LEN
 ::= INT_TYPE_SYM
 ::= DBL_TYPE_SYM
 ::= DATE_TYPE_SYM
 ::= DATETIME_TYPE_SYM
 ::= TYPE_ID
 ::= QUEUE_TYPE_SYM
 ::= VOID_TYPE_SYM
 ::= ANY_TYPE_SYM
 ::= GUID_TYPE_SYM
 ::= INT64_TYPE_SYM
 ::= CLR_TYPE
CLR_TYPE ::= CLR_NAMESPACE TYPE_ID CLR_ARRAY_TYPE_EXT
 ::= CLR_NAMESPACE CLR_TYPE
CLR_NAMESPACE ::= TYPE_ID PERIOD_SYM
CLR_ARRAY_TYPE_EXT ::= CLR_ARRAY_SPEC
 ::= CLR_ARRAY_SPEC CLR_ARRAY_PART
CLR_ARRAY_PART ::= CLR_ARRAY_LEFT_PART CLR_RECTANGULAR_LIST RGHT_BRKT_SYM
CLR_ARRAY_LEFT_PART ::= LEFT_BRKT_SYM
CLR_RECTANGULAR_LIST ::= CLR_COMMA_LIST
 ::= LIST_SEP_SYM
 ::= CLR_COMMA_LIST LIST_SEP_SYM
STR_LEN ::= INT_SYM
 ::= ARR_DCL_IDX LEFT_BRKT_SYM RANGE ARRAY_MEM RGHT_BRKT_SYM
RANGE ::= IF_EXPR
 ::= ARRAY_MEM LIST_SEP_SYM IF_EXPR
 ::= ASG_CLAUSE INIT_START IF_EXPR
 ::= INIT_START ASG_SYM
ASG_STMT ::= LVAL_FLD ASSIGN IF_EXPR
 ::= LVAL_LIST ASG_SYM IF_EXPR
 ::= LVAL_FLD ASG_INC_DEC
 ::= ASG_INC_DEC LVAL_FLD
 ::= LVAL_FLD ASG_EVENT_HANDLER
ASIGN ::= ASG_SYM
 ::= ASGINC_SYM
 ::= ASGDEC_SYM
ASG_INCDEC ::= ASGINC_SYM
 ::= ASGDEC_SYM
ASG_EVENT_HANDLER ::= ASG_INCDEC EVENTHANDLER_SYM LEFT_PAR_SYM QUALIFIER STD_ID RGHT_PAR_SYM
 ::= ASG_INCDEC EVENTHANDLER_SYM LEFT_PAR_SYM STD_ID DBLCOLON_SYM STD_ID RGHT_PAR_SYM
 ::= ASG_INCDEC EVENTHANDLER_SYM LEFT_PAR_SYM QUALIFIER EVAL_CLR_TYPE DBLCOLON_SYM STD_ID RGHT_PAR_SYM
ASG_INC_DEC ::= INC_SYM
 ::= DEC_SYM
CLR_EMPTY_COMMA_LIST ::= CLR_EMPTY_RECT_COMMA_LIST
::=
CLR_EMPTY_COMMA_LIST ::= LIST_SEP_SYM
    ::= CLR_EMPTY_RECT_COMMA_LIST LIST_SEP_SYM
CONLITTERAL ::= LEFT_BRKT_SYM IF_EXPR EXPR_LIST RGHT_BRKT_SYM
CONSTANT ::= INT_SYM
    ::= DBL_SYM
    ::= STR_SYM
    ::= DATE_SYM
    ::= DATETIME_SYM
    ::= STD_ID DBLCOLON_SYM STD_ID
    ::= TRUE_SYM
    ::= FALSE_SYM
    ::= NULL_SYM
    ::= INT64_SYM
    ::= QUALIFIER EVAL_CLR_TYPE DBLCOLON_SYM STD_ID
    ::= QUALIFIER STD_ID DBLCOLON_SYM STD_ID
DIRSEARCH ::= DIRS_HEADER PERIOD_SYM STD_ID ARR_IDX
    ::= DIRS_HEADER PERIOD_SYM FLD_NUM ARR_IDX
DIRS_HEADER ::= LEFT_PAR_SYM SET_DIRS FIND_JOIN RGHT_PAR_SYM
SET_DIRS ::= FIELD
    ::= QUALIFIER STD_ID ARR_IDX
    ::= QUALIFIER FLD_NUM ARR_IDX
    ::= STD_ID ARR_IDX
QUALIFIER ::= EVAL PERIOD_SYM
    ::= STD_ID PERIOD_SYM
FLD_NUM ::= LEFT_PAR_SYM IF_EXPR RGHT_PAR_SYM
ARR_IDX ::= LEFT_BRKT_SYM SMPL_EXPR RGHT_BRKT_SYM
EXPR_LIST ::= EXPR_LIST2
    ::= EXPR_LIST2 LIST_SEP_SYM IF_EXPR
EXPR_LIST2 ::= LIST_SEP_SYM IF_EXPR
    ::= EXPR_LIST2 LIST_SEP_SYM IF_EXPR
FUNCTION ::= FUNC_ID LEFT_PAR_SYM EVAL_FUNCTION_NAME PAR_LIST RGHT_PAR_SYM
EVAL_FUNCTION_NAME ::= EVAL_NAME
    ::= EVAL_NAME PAR_LIST RGHT_PAR_SYM
    ::= EVAL_ID LEFT_PAR_SYM
    ::= STD_ID LEFT_PAR_SYM
    ::= STD_ID DBLCOLON_SYM STD_ID LEFT_PAR_SYM
    ::= SUPER_SYM LEFT_PAR_SYM
    ::= NEW_SYM STD_ID LEFT_PAR_SYM
    ::= NEW_SYM EVAL_CLR_TYPE LEFT_PAR_SYM
    ::= QUALIFIER EVAL_CLR_TYPE DBLCOLON_SYM STD_ID LEFT_PAR_SYM
    ::= QUALIFIER STD_ID LEFT_PAR_SYM
    ::= QUALIFIER STD_ID DBLCOLON_SYM STD_ID LEFT_PAR_SYM
EVAL_CLR_TYPE ::= NAMESPACE STD_ID
    ::= NAMESPACE EVAL_CLR_TYPE
NAMESPACE ::= STD_ID PERIOD_SYM
EVAL ::= EVAL_NAME PAR_LIST RGHT_PAR_SYM
PAR_LIST ::= PRM_LIST
    ::= PRM_LIST LIST_SEP_SYM PRM_LIST
PRM_LIST ::= PAR_ELEM
    ::= PRM_LIST LIST_SEP_SYM PAR_ELEM
PAR_ELEM ::= IF_EXPR
    ::= BYREF_SYM FIELD
INTRINSICS ::= INTRI_ID LEFT_PAR_SYM IARGS RGHT_PAR_SYM
IARGS ::= STD_ID
    ::= STR_SYM
    ::= STD_ID LIST_SEP_SYM STD_ID
::=
STMTLIST ::= STATEMENTS
::=
STATEMENTS ::= STATEMENT
    ::= STATEMENTS STATEMENT
STATEMENT ::= COMPOUND_STMT
    ::= WHILE_STMT
    ::= FOR_STMT
    ::= DO_STMT
    ::= SEARCH_STMT
    ::= FIND_STMT
SEARCH_JOIN ::= SEARCH_WHERE JOIN_LIST
SEARCH_WHERE ::= SEARCH_ORDER WHERE IF_EXPR
::= SEARCH_ORDER
WHERE ::= WHERE_SYM
SUM_ELEM ::= SUM_FUNC LEFT_PAR_SYM STD_ID RIGHT_PAR_SYM
SUM_FUNC ::= SUM_SYM
::= AVG_SYM
::= CNT_SYM
::= MINOF_SYM
::= MAXOF_SYM
SEARCH_ORDER ::= SEARCH_USING
::= SEARCH_USING ORDER_GROUP
ORDER_GROUP ::= ORDERBY_CLAUSE OPT_GROUPBY
::= GROUPBY_CLAUSE OPT_ORDERBY
OPT_GROUPBY ::= GROUPBY_CLAUSE
::=
OPT_ORDERBY ::= ORDERBY_CLAUSE
::=
ORDERBY_CLAUSE ::= ORDER_SYM OPT_BY ORDER_ELEM
::= ORDER_SYM LIST_SEP_SYM ORDER_ELEM
GROUPBY_CLAUSE ::= GROUP_SYM OPT_BY ORDER_ELEM
::= GROUP_SYM LIST_SEP_SYM ORDER_ELEM
ORDER_ELEM ::= STD_ID INDEX DIRECTION
::= ORDER_QUALIFIER STD_ID INDEX DIRECTION
ORDER_QUALIFIER ::= STD_ID PERIOD_SYM
INDEX ::= LEFT_BRKT_SYM INT_SYM RIGHT_BRKT_SYM
::=
DIRECTION ::= ASCEND_SYM
::=
DESCEND_SYM
::=
OPT_BY ::= BY_SYM
::=
SEARCH_USING ::= SEARCH_CLAUSE USING_INDEX STD_ID
::= SEARCH_CLAUSE USING_INDEX HINT_SYM STD_ID
::= SEARCH_CLAUSE
USING_INDEX ::= INDEX_SYM
SEARCH_CLAUSE ::= WHILE_SYM SELECT_SYM SELECTOPT CROSSCOMPANY_CLAUSE VALIDTIMESTATE_CLAUSE TABLE
CROSSCOMPANY_CLAUSE ::= CROSSCOMPANY_SYM
::= CROSSCOMPANY_SYM COLON_SYM STD_ID
::=
VALIDTIMESTATE_CLAUSE ::= VALIDTIMESTATE_SYM LEFT_PAR_SYM STD_ID LIST_SEP_SYM STD_ID RIGHT_PAR_SYM
::= VALIDTIMESTATE_SYM LEFT_PAR_SYM STD_ID RIGHT_PAR_SYM
::=
SELECTOPT ::= SELECTOPT
::= SELECTOPT REVERSE_SYM
::= SELECTOPT FIRSTFAST_SYM
::= SELECTOPT FIRSTONLY_SYM
::= SELECTOPT FIRSTONLY_SYM1
::= SELECTOPT FIRSTONLY_SYM10
::= SELECTOPT FIRSTONLY_SYM100
::= SELECTOPT FIRSTONLY_SYM1000
::= SELECTOPT FORUPDATE_SYM
::= SELECTOPT NOFETCH_SYM
::= SELECTOPT FORCE_SELECT_ORDER_SYM
::= SELECTOPT FORCE_NESTED_LOOP_SYM
::= SELECTOPT FORCE_LITERALS_SYM
::= SELECTOPT FORCE_PLACEHOLDERS_SYM
::= SELECTOPT REPEATTABLEREAD_SYM
::= SELECTOPT OPTIMISTICLOCK_SYM
::= SELECTOPT PESSIMISTICLOCK_SYM
::= SELECTOPT GENERATEONLY_SYM
FIND_STMT ::= FIND_JOIN SEMICOLON_SYM
FIND_JOIN ::= FIND_WHERE JOIN_LIST
FIND_WHERE ::= FIND_ORDER WHERE IF_EXPR
::= FIND_ORDER
FIND_ORDER ::= FIND_USING
::= FIND_USING ORDER_GROUP
FIND_USING ::= FIND_TABLE USING_INDEX STD_ID
::= FIND_TABLE USING_INDEX HINT_SYM STD_ID
FIND_TABLE ::= SELECT_SYM SELECTOPT CROSSCOMPANY_CLAUSE VALIDTIMESTATE_CLAUSE TABLE
DELETE_SYM SELECTOPT CROSSCOMPANY_CLAUSE VALIDTIMESTATE_CLAUSE TABLE
TABLE ::= FLD_LIST OPT_FROM
FLD_LIST ::= MULT_SYM
FIELD_LIST ::= FIELD_SPEC
FIELD_SPEC ::= STD_ID INDEX
OPT_FROM ::= FROM_SYM STD_ID

UPDATE_STMT ::= UPDATETABLE SET_SYM SETFIELDSMODE FIELDASSIGNMENTS OPT_WHERE JOIN_LIST
SEMICOLON_SYM

UPDATETABLE ::= UPDATE_SYM SELECTOPT CROSSCOMPANY_CLAUSE STD_ID
OPT_WHERE ::= WHERE IF_EXPR
FIELDASSIGNMENTS ::= FIELDASSIGNMENTS LIST_SEP_SYM FIELDASSIGNMENT
FIELDASSIGNMENT ::= STD_ID INDEX ASG_SYM IF_EXPR
INSERT_PART ::= INSERT_SYM CROSSCOMPANY_CLAUSE INSERT_NAME LEFT_PAR_SYM INSERTFIELDLIST
RIGHT_PAR_SYM
INSERT_NAME ::= STD_ID
INSERT_STMT ::= INSERT_PART FIND_JOIN SEMICOLON_SYM
INSERTFIELDLIST ::= INSERTFIELD
INSERTFIELD ::= INSERTFIELDLIST LIST_SEP_SYM INSERTFIELD
PRINT_STMT ::= PRINT_CLAUSE AT_CLAUSE SEMICOLON_SYM
PRINT_CLAUSE ::= PRINT IF_EXPR EXPR_LIST
PRINT_SYM
AT_CLAUSE ::= AT_SYM IF_EXPR LIST_SEP_SYM IF_EXPR
WINDOW_STMT ::= WINDOW_SYM IF_EXPR LIST_SEP_SYM IF_EXPR AT_CLAUSE SEMICOLON_SYM
IF_STMT ::= ELSE_STMT
ELSE ::= IF_CONDS ELSE_SYM
SWITCH_STMT ::= CASE_LIST RIGHTBR_SYM
CASE_LIST ::= SWITCH_SYM LEFT_PAR_SYM IF_EXPR RGHT_PAR_SYM LEFTBR_SYM
CASE_TESTS ::= CASE_HEADER COLON_SYM
CASE_HEADER ::= CASE IF_EXPR
CASEALT ::= CASE_HEADER LIST_SEP_SYM
CASE_STMT ::= ASG_STMT SEMICOLON_SYM
FUNCTION SEMICOLON_SYM
INTRINSICS SEMICOLON_SYM
EVAL SEMICOLON_SYM
PAUSE_STMT ::= PAUSE_SYM SEMICOLON_SYM
BP_CLAUSE ::= BP_SYM SEMICOLON_SYM
BREAK_STMT ::= BREAK_SYM SEMICOLON_SYM
CONTINUE_STMT ::= CONTINUE_SYM SEMICOLON_SYM
RETURN_CLAUSE ::= RETURN_SYM SEMICOLON_SYM
RETURN_SYM IF_EXPR SEMICOLON_SYM
TTS_STMT ::= TTSABORT_SYM SEMICOLON_SYM
TTSBEGIN_SYM SEMICOLON_SYM
TTSEND_SYM SEMICOLON_SYM
FLUSH_STMT ::= FLUSH ID_LIST SEMICOLON_SYM
FLUSH_SYM
TBLLOCK_STMT ::= TABLELOCK ID_LIST SEMICOLON_SYM
TABLELOCK_SYM
ID_LIST ::= STD_ID
ID_LIST ::= ID_LIST LIST_SEP_SYM STD_ID
STRING RFC_STMT ::= RFC_SYM TARI F SEMICOLON_SYM
X++ Language Syntax is Stricter in Microsoft Dynamics AX 2012

Starting in Microsoft Dynamics AX 2012, the syntax rules for X++ are stricter than in previous versions of the product. This topic describes the syntax changes.

Table of X++ Syntax Changes

The following table displays a list of syntax changes that start in Microsoft Dynamics AX 2012.

<table>
<thead>
<tr>
<th>AREA</th>
<th>SYNTAX RULE</th>
<th>BEFORE MICROSOFT DYNAMICS AX 2012</th>
<th>STARTING WITH MICROSOFT DYNAMICS AX 2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>Escape</td>
<td>The backslash character \ is rejected by the compiler for unrecognized escapes</td>
<td>The compiler used to accept &quot;31\12\2002&quot;, but during runtime the literal string was interpreted as a different value.</td>
<td>Now the following X++ statement is rejected by the compiler: str myDateString = &quot;31\12\2002&quot;; The proper syntax is &quot;31\12\2002&quot;.</td>
</tr>
<tr>
<td>Exceptions</td>
<td>Retry is no longer allowed outside of a catch block</td>
<td>It was possible to write the retry keyword outside of a catch block. This caused the program to end when the retry was reached during runtime.</td>
<td>Now retry can occur only inside a catch block. For more information, see Exception Handling with try and catch Keywords.</td>
</tr>
<tr>
<td>Exceptions</td>
<td>Now you can throw and catch only int values</td>
<td>It was possible to throw scalar expressions like strings and dates, such as throw &quot;hello world&quot;; and get no compile error. At runtime this was catch-able by a catch block that was not decorated with any specific value, such as catch {print(&quot;Catch worked.&quot;);}.</td>
<td>Now the only expression you can put on the throw keyword is an int. Often the best thing to throw is Global.error(&quot;Explanation&quot;);. Often the best thing to catch is an element of the Exception enum. For more information, see Exception Handling with try and catch Keywords.</td>
</tr>
<tr>
<td>AREA</td>
<td>SYNTAX RULE</td>
<td>BEFORE MICROSOFT DYNAMICS AX 2012</td>
<td>STARTING WITH MICROSOFT DYNAMICS AX 2012</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Inheritance</td>
<td>Downcasting can now be explicit.</td>
<td>It was possible to assign a base object to a derived object with the simple assignment operator, which is the equals sign (=). The compiler accepted these assignments, but during run time any misuse of an improper downcast assignment caused an error.</td>
<td>Now all downcasts can be explicit. This is accomplished with the new as expression operator. Explicit downcasting with the as keyword is illustrated by the following code example, in which ThingClass extends Object:</td>
</tr>
<tr>
<td></td>
<td>NOTE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>It is good programming practice to avoid implicit downcasts.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inheritance</td>
<td>Override of a base method cannot be less accessible than the base method</td>
<td>It was possible to have a base method be decorated with protected and yet have an override of that method be private.</td>
<td>Now when a base method is protected, the override method must be either protected or public, and the override method cannot be private. For more information, see Method Access Control.</td>
</tr>
<tr>
<td>Inheritance</td>
<td>Override of a base method must have the exact same return type and parameter signature as the base method</td>
<td>Suppose a base class had a method that inputs a parameter of the Common table, which is the base of all tables. In a derived class it was possible to override the method to instead input MyTable.</td>
<td>Now the parameter signatures of the base method and its override method must match exactly. Also, the return types must match exactly. For more information, see Overriding a Method.</td>
</tr>
<tr>
<td>Interfaces</td>
<td>Implementation of an interface method must match the parameter signature exactly</td>
<td>Suppose an interface had a method that input a parameter of an int. In a class that implements the interface, it was possible to write the method with a parameter of a str.</td>
<td>Now the parameter signatures of the method must exactly match between the interface and the implementation of the method on a class. Also, the return types must match exactly. For more information, see Interfaces Overview.</td>
</tr>
<tr>
<td>AREA</td>
<td>SYNTAX RULE</td>
<td>BEFORE MICROSOFT DYNAMICS AX 2012</td>
<td>STARTING WITH MICROSOFT DYNAMICS AX 2012</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Interfaces</td>
<td>A non-abstract base class that implements an interface cannot rely on a derived class for that implementation</td>
<td>When a base class implements an interface, it was possible for the class to not implement the methods of the interface if a derived class implemented the methods. The only limitation was that the new constructor method could not be called on the class.</td>
<td>Now the compiler requires that every class that implements an interface must have or inherit a complete implementation of every method of the interface. For more information, see X++, C# Comparison: Object Oriented Programming.</td>
</tr>
<tr>
<td>Modifiers</td>
<td>The static modifier should not be applied to an interface</td>
<td>It was possible to write static interface IMyInterface {}, but the static modifier had no effect because it makes no sense in this context.</td>
<td>Sometime after Dynamics AX 2009 the X++ compiler might stop allowing the static modifier on interface declarations. For more information, see Interfaces Overview.</td>
</tr>
<tr>
<td>Modifiers</td>
<td>The static modifier must not be applied to the new constructor</td>
<td>It was possible to apply the static modifier to the declaration of the new constructor method. This caused new MyClass(); to behave as a null operation. Instead, the statement MyClass::new(); would call the static new method, but that would not construct an object.</td>
<td>Now the compiler issues an error when the static modifier is applied to the new method. For more information, see Constructors.</td>
</tr>
<tr>
<td>Modifiers</td>
<td>Use an explicit access modifier on each method</td>
<td>In the past the menu item of AOT &gt; Classes &gt; MyClass &gt; New Method created the method without any access modifier. This meant that the method was implicitly public, although some X++ developers might not have been fully aware of the default. This created extra work later when a developer needed to modify the code in the method, because the developer had to research everywhere that the method might be called from.</td>
<td>Now the New Method menu item explicitly includes the private keyword in its automatic declaration of the new method. The developer can type in a different modifier if appropriate. For more information, see Method Modifiers.</td>
</tr>
</tbody>
</table>
## Parameters

<table>
<thead>
<tr>
<th>SYNTAX RULE</th>
<th>BEFORE MICROSOFT DYNAMICS AX 2012</th>
<th>STARTING WITH MICROSOFT DYNAMICS AX 2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>Parameters given in a call to a \texttt{new} constructor method must match the parameters on the \texttt{new} constructor method.</td>
<td>It was possible to pass in multiple parameters on call to a \texttt{new} constructor method even when the \texttt{new} method was declared to input no parameters.</td>
</tr>
</tbody>
</table>

### Parameters

Parameters with default values must come after all parameters that do not have default values.

It was possible to declare a method that takes in two parameters, and have only the first parameter offer a default value. There was no purpose to this. There was no way to accept the default of the first parameter because the call must specify a value for the second parameter and cannot omit the first parameter.

Now in the declaration of a method, any parameter that offers a default value must come after all the parameters that do not. For more information, see the following topics:
- Using Optional Parameters
- Best Practices for Parameters

### Parameters

Override of a method must have the same default parameters as the overridden method.

It was possible to declare a method as `public void myMethod(int i=22){}` and the override as `public void myMethod(){}`. But if the override method was called as `derivedObject(333);` an error occurred.

Now the override method must list the same parameter types in the same sequence that they are declared in the overridden method. For more information, see Overriding a Method.

### Preprocessor

A \texttt{TODO} in a comment must be the first non-whitespace in the first line of the comment.

The X++ preprocessor used to detect the \texttt{TODO} keyword in a multi-line `/* ... */` task comment even when the \texttt{TODO} appeared after other text after the first comment line.

Now the X++ preprocessor detects the \texttt{TODO} keyword only if \texttt{TODO} appears on the first line of the comment, and as the first non-whitespace in the comment. For more information, see \texttt{TODO} Comments for X++ Developer Tasks.

---

### Additional resources

- **X++ Language Reference**

---
This topic describes where to find API documentation in Visual Studio and on the Microsoft docs site.

Application classes and tables

**Application class and table documentation is in Visual Studio**

You can find documentation for the Application classes in Microsoft Visual Studio. Search for the class name in Application Explorer and then display the code. You can find additional metadata about the class in the Properties window. You can download a list of all the tables in the Technical Reference Reports. For more information, see Find information about standard data entities.

**Programming with application tables and classes**

Application tables are similar to application classes, but with the following differences from classes:

- Tables are persistent.
- Table fields are always public.
- A table almost always corresponds to a real object.
- The definition of a table must sometimes be erased if you later want another table to extend it.

**Design pattern of private new in application classes**

All application classes are under Application Explorer > Classes. Every application class has the constructor method named `new`, even if the class has no `new` node in the Application Explorer. If the class has no explicit `new` node, the implicit `new` method is public. A design pattern that is sometimes used in the application classes is to declare the explicit `new` constructor method as `private`. Then a `public static` method is added to call the `new` method. The static method can restrict or control the call the `new` method based on various conditions, if necessary.

System classes and tables

**System API, class, and table documentation is on the Microsoft docs site**

Documentation for the classes and functions that are listed under System Documentation in Application Explorer is available on the Microsoft docs site.

**X++ compile-time functions**

- X++ compile-time functions

**X++ run-time functions**

- X++ run-time functions:
  - X++ container run-time functions
  - X++ business run-time functions
  - X++ conversion run-time functions
  - X++ date run-time functions
  - X++ math run-time functions
  - X++ reflection run-time functions
System tables

System classes

The reference documentation for the system classes is in the .NET API browser.

API reference for finance and operations apps

Microsoft.Dynamics.Ax.Xpp namespace

Dynamics.AX.Application namespace
Overview

Compile-time functions are executed early during compilation of X++ code. They should be used wherever possible in X++ code to make the code resilient to changes to the metadata stored in the Application Explorer. Compile-time functions have their input value verified by the compiler. If the input value is not found to match any existing object in the Application Explorer, the compiler issues an error. The inputs to these functions must be literals, because the compiler cannot determine the value that a variable contains at run time. A compile-time function is a metadata assertion function. It takes arguments that represents an entity in the Application Explorer and validates the arguments at compile time. It has no effect at run time. Attributes are classes that inherit from the `SysAttribute` class. To support the validation of form, report, query, and menu metadata, use the `AutoDeclaration` property on controls. Most of these functions retrieve metadata about items that are in the Application Explorer. Some common compile time functions are as follows:

- `classNum` – Retrieves the ID of a class.
- `classStr` – During compile time, verifies that a class of that name exists. This approach is better than discovering the error later during run time.
- `evalBuf` – Evaluates the input string of X++ code, and then returns the results as a string.
- `literalStr` – retrieves a label ID when given the string representation of a label, such as the string `"@SYS12345"`. For example, `myLabel.exists(literalStr("@SYS12345"));`.

**NOTE**

X++ compile time functions cannot be called from a .NET program.

Functions

**attributeStr**

Validates that the specified attribute class exists in the Application Explorer; if not, a compiler error occurs.

**Syntax**

```
str classStr(class class)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>The name of the attribute to validate.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the attribute.

**Remarks**
This is a compile-time function. For more information, see Overview.

Example

```java
static void attributeStrExample(Args _args)
{
    str s;
    s = attributeStr(AifDocumentOperationAttribute);
    print s;
    pause;
}
```

classNum

Retrieves the ID of the specified class.

Syntax

```java
int classNum(class class)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>The class for which to retrieve the ID.</td>
</tr>
</tbody>
</table>

Return Value

The ID of the specified class.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```java
static void classNumExample(Args _args)
{
    int i;
    i = classNum(Global);
    print i;
    pause;
}
```

classStr

Retrieves the name of a class as a string.

Syntax

```java
str classStr(class class)
```

Parameters
### class

The name of the class to return.

**Return Value**

The name of the class.

**Remarks**

Use this function instead of literal text to retrieve a string that contains the class name. If the class does not exist, the function generates a syntax error at compile time. This is a compile-time function. For more information, see [Overview](#).

**Example**

```c
static void clStrExample(Args _args)
{
    str s;
    s = classStr(Global);
    print s;
    pause;
}
```

### configurationKeyNum

Retrieves the ID of the specified configuration key.

**Syntax**

```c
int configurationKeyNum(str keyname)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>keyname</td>
<td>The configuration key for which to return the ID.</td>
</tr>
</tbody>
</table>

**Return Value**

The ID of the specified configuration key.

**Remarks**

This is a compile-time function. For more information, see [Overview](#).

**Example**

```c
static void configurationKeyNum(Args _args)
{
    int i;
    i = configurationKeyNum(AIF);
    print i;
    pause;
}
```

### configurationKeyStr

The name of the class.

Use this function instead of literal text to retrieve a string that contains the class name. If the class does not exist, the function generates a syntax error at compile time. This is a compile-time function. For more information, see [Overview](#).
Retrieves the name of a configuration key as a string.

**Syntax**

```c
str configurationKeyStr(str keyname)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>keyname</td>
<td>The name of the configuration key.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the configuration key.

**Remarks**

Use this function instead of literal text to retrieve a string that contains the configuration key name. If the key does not exist, the function generates a syntax error at compile time. This is a compile-time function. For more information, see [Overview](#).

**Example**

```c
static void configurationKeyStrExample(Args _args)
{
    str s;
    s = configurationKeyStr(AIF);
    print s;
    pause;
}
```

dataEntityDataSourceStr

Retrieves the name of a data source of a data entity.

**Syntax**

```c
str dataEntityDataSourceStr(str dataEntity, str dataSource)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataEntity</td>
<td>The name of the data entity.</td>
</tr>
<tr>
<td>dataSource</td>
<td>The name of the data source.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the data source.

**Remarks**

This is a compile-time function. For more information, see [Overview](#).

**Example**

No example.
delegateStr

Returns the name of the delegate.

Syntax

```c
str delegateStr(str class, str instanceDelegate)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>The name of the class, table, or form.</td>
</tr>
<tr>
<td>instanceDelegate</td>
<td>The name of the instance delegate.</td>
</tr>
</tbody>
</table>

Return Value

The name of the delegate.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

dimensionHierarchyLevelStr

Returns the name of the dimension hierarchy level.

Syntax

```c
str dimensionHierarchyLevelStr(str dimensionHierarchyLevel)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>dimensionHierarchyLevel</td>
<td>The name of the dimension hierarchy level.</td>
</tr>
</tbody>
</table>

Return Value

The name of the dimension hierarchy level.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

dimensionHierarchyStr

Returns the name of the dimension hierarchy.

Syntax
str dimensionHierarchyStr(str dimensionHierarchy)

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>dimensionHierarchy</td>
<td>The name of the dimension hierarchy.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the dimension hierarchy.

**Remarks**

This is a compile-time function. For more information, see [Overview](#).

**Example**

No example.

dimensionReferenceStr

Returns the name of the dimension reference.

**Syntax**

str dimensionReferenceStr(str dimensionReference)

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>dimensionReference</td>
<td>The name of the dimension reference.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the dimension reference.

**Remarks**

This is a compile-time function. For more information, see [Overview](#).

**Example**

No example.

dutyStr

Retrieves a string that represents the name of the specified security duty.

**Syntax**

str dutyStr(str securityDuty)

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>securityDuty</td>
<td>The name of the security duty.</td>
</tr>
</tbody>
</table>
**Return Value**
The name of the security duty in a string.

**Remarks**
This is a compile-time function. For more information, see *Overview*.

**Example**
No example.

### enumCnt
Retrieves the number of elements in the specified enumeration type.

**Syntax**
```
int enumCnt(enum enumtype)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>enumtype</td>
<td>The enumeration type.</td>
</tr>
</tbody>
</table>

**Return Value**
The number of elements in the specified enumeration type.

**Remarks**
This is a compile-time function. For more information, see *Overview*.

**Example**
```
enumCnt(NoYes); //Returns 2, as the two elements are Yes and No.
```

### enumLiteralStr
Indicates whether the specified string is an element of the specified enumeration type.

**Syntax**
```
\enumLiteralStr(enum enum, string str)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td>The enumeration type from which to retrieve the specified value.</td>
</tr>
</tbody>
</table>

**Return Value**
The value of the *str* parameter if the specified string was found; otherwise, a compilation error.

**Remarks**
This is a compile-time function. For more information, see *Overview*.
Example

```c
static void getEnumValueAsString()
{
    str i;
    i = enumLiteralStr(ABCEnum, "valueInABCEnum");
    print i;
    pause;
}
```

denumNum

Retrieves the ID of the specified enumeration type.

**Syntax**

```c
int enumNum(enum enum)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td>The enumeration for which to return the ID.</td>
</tr>
</tbody>
</table>

**Return Value**

The ID of the specified enumeration type.

**Remarks**

This is a compile-time function. For more information, see [Overview](#).

Example

```c
static void enumNum(Args _args)
{
    int i;
    i = enumNum(ABC);
    print i;
    pause;
}
```

denumStr

Retrieves the name of an enumeration as a string.

**Syntax**

```c
str enumStr(enum enum)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td>The name of the enumeration.</td>
</tr>
</tbody>
</table>

**Return Value**
The name of the enumeration.

**Remarks**
This is a compile-time function. For more information, see [Overview](#).

**Example**

```cpp
static void enumStrExample(Args _args)
{
    str s;
    s = enumStr(ABC);
    print s;
    pause;
}
```

### extendedTypeNum

Retrieves the ID of the specified extended data type.

**Syntax**

```cpp
int extendedTypeNum(int str)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>str</td>
<td>The extended data type for which to return the ID.</td>
</tr>
</tbody>
</table>

**Return Value**

The ID of the specified extended data type.

**Remarks**
This is a compile-time function. For more information, see [Overview](#).

**Example**

```cpp
static void EDTNum(Args _args)
{
    int i;
    str s;
    
    i = extendedTypeNum(AccountName);
    s = extendedTypeStr(AccountName);
    print  int2Str(i);
    print  s;
    pause;
}
```

### extendedTypeStr

Retrieves the name of an extended data type as a string.

**Syntax**

```cpp
```
str extendedTypeStr(int str)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>str</td>
<td>The name of the extended data type.</td>
</tr>
</tbody>
</table>

Return Value

The name of the extended data type.

Remarks

Use this function instead of literal text to return a string that contains the extended data type name. If the data type does not exist, the `extendedTypeStr` function generates a syntax error at compile time. This is a compile-time function. For more information, see Overview.

Example

```java
static void EDTStr(Args _args)
{
    int i;
    str s;
    i = extendedTypeNum(AccountName);
    s = extendedTypeStr(AccountName);
    print  int2Str(i);
    print  s;
    pause;
}
```

fieldNum

Returns the ID number of the specified field.

Syntax

```java
int fieldNum(str tableName, str fieldName)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>The name of the table.</td>
</tr>
<tr>
<td>fieldName</td>
<td>The name of the field.</td>
</tr>
</tbody>
</table>

Return Value

The ID of the specified field.

Remarks

This is a compile-time function. For more information, see Overview.

Example

The following example prints the number of the CashDisc field in the CustTable table.
static void fieldNumExample(Args _args)
{
    int myInt;
    
    myInt = fieldNum(CustTable, CashDisc);
    Global::info(strfmt("CashDisc has a field ID of %1 in the CustTable table.", myInt));
}
/***Infolog Display
Message (10:40:00 am)
CashDisc has a field ID of 10 in the CustTable table.
****/

fieldPName

Retrieves the label of the specified field.

Syntax

str fieldPName(str tableid, str fieldid)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableid</td>
<td>The table that contains the specified field.</td>
</tr>
<tr>
<td>fieldid</td>
<td>The field to convert.</td>
</tr>
</tbody>
</table>

Return Value

The label of the field.

Remarks

This is a compile-time function. For more information, see Overview.

Example

The following example prints the label of the CashDisc field.

static void fieldPNameExample(Args _arg)
{
    str myText;
    
    myText = fieldPName(CustTable, CashDisc);
    Global::info(strfmt("%1 is the label of the CashDisc field.", myText));
}
/***Infolog Display
Message (02:00:57 pm)
Cash discount is the label of the CashDisc field.
****/

fieldStr

Retrieves the field name of the specified field.

Syntax
**fieldStr**

```c
str fieldStr(str tableid, str fieldid)
```

### Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableid</td>
<td>The table that contains the field.</td>
</tr>
<tr>
<td>fieldid</td>
<td>The field to convert.</td>
</tr>
</tbody>
</table>

**Return Value**

The field name of the specified field.

**Remarks**

This is a compile-time function. For more information, see [Overview](#).

**Example**

The following example assigns the name of the `CashDisc` field to the `myText` variable.

```c
static void fieldStrExample(Args _arg)
{
    str myText;

    myText = fieldStr(CustTable, CashDisc);
    Global::info(strfmt("%1 is the specified field.", myText));
}
```

### FormControlStr

**Syntax**

```c
str formControlStr(formName, controlName)
```

#### Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>formName</td>
<td>The name of the form, not in quotation marks.</td>
</tr>
<tr>
<td>controlName</td>
<td>The name of the control that is on the form, not in quotation marks.</td>
</tr>
</tbody>
</table>

**Return Value**

A string that contains the name of the control as it appears in the Application Explorer.

**Remarks**

Causes the X++ compiler to check whether the control exists on the form, and to replace the function call with a string of the valid control name.
formDataFieldStr

Returns the name of a data field in a form.

Syntax

```plaintext
str formDataFieldStr(str formName, str dataSource, str dataField)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>formName</td>
<td>The name of the form.</td>
</tr>
<tr>
<td>dataSource</td>
<td>The data source of the form.</td>
</tr>
<tr>
<td>dataField</td>
<td>The data field of the data source.</td>
</tr>
</tbody>
</table>

Return Value

The name of a data field in a form.

Remarks

This is a compile-time function. For more information, see Overview.

Example

```
str a = formDataFieldStr(FMVehicle, FMModelRate, RatePerDay);
```

formDataSourceStr

Returns the name of a data source in a form.

Syntax

```plaintext
str formDataSourceStr(str formName, str dataSource)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>formName</td>
<td>The name of the form.</td>
</tr>
</tbody>
</table>

A compile error is issued if the compiler determines that the control does not exist on the form. If your X++ code uses a string that contains quotation marks to supply the control name, the error cannot be discovered until run time. Use of this function enables the compiler to discover the error earlier at compile time. X++ functions such as `formControlStr` that are executed by the compiler are called compile-time functions or compile-time functions. That is why the input parameters are not standard strings in quotation marks. Compile-time functions are not represented in the p-code or other executable that is output by the compiler. This is a compile-time function. For more information, see Overview.

Example

No example.
<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataSource</td>
<td>The data source of the form.</td>
</tr>
</tbody>
</table>

**Return Value**  
The name of a data source in a form.

**Remarks**  
This is a compile-time function. For more information, see *Overview*.

**Example**

```c
str b = formDataSourceStr(FMVehicle, FMModelRate);
```

**formMethodStr**

Returns the name of a method of a form.

**Syntax**

```c
str formMethodStr(str formName, str methodName)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>formName</td>
<td>The name of the form.</td>
</tr>
<tr>
<td>methodName</td>
<td>The method of the form.</td>
</tr>
</tbody>
</table>

**Return Value**  
The name of a method in a form.

**Remarks**  
This is a compile-time function. For more information, see *Overview*.

**Example**

The following example prints the name of the `showDialog` method.

```c
str c = formMethodStr(Batch, showDialog);
```

**formStr**

Retrieves the name of a form.

**Syntax**

```c
str formStr(str form)
```

**Parameters**
### form

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>form</td>
<td>The name of a form.</td>
</tr>
</tbody>
</table>

**Return Value**

A string that represents the name of the form.

**Remarks**

This is a compile-time function. For more information, see Overview.

**Example**

The following example prints the name of the InventDim form.

```plaintext
static void formStrExample(Args _arg)
{
    
    Global::info(formStr(InventDim));
}
/****Infolog Display
Message (11:04:39 am)
InventDim
****/
```

#### identifierStr

Converts the specified identifier to a string.

**Syntax**

```plaintext
str identifierStr(str ident)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ident</td>
<td>The identifier to convert.</td>
</tr>
</tbody>
</table>

**Return Value**

A string that represents the specified identifier.

**Remarks**

You will receive a best practice warning if you use the `identifierStr` function. This occurs because existence checking is performed for `identifierStr`. Try to use a more specific compile-time function if one is available. This is a compile-time function. For more information, Overview.

**Example**

The following code example assigns the `myvar` variable name to the `thevar` variable.
static void identifierStrExample(Args _args)
{
    str myvar;
    str thevar
    ;

    thevar = "[" + identifierStr(myvar) + "]";
    Global::info(strfmt(thevar));
} /**Infolog Display
Message (09:19:49 am)
[myvar]
*****/

indexNum

Converts the specified index to a number.

Syntax

int indexNum(str tableid, str indexid)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableid</td>
<td>The table that contains the index.</td>
</tr>
<tr>
<td>indexid</td>
<td>The index to convert.</td>
</tr>
</tbody>
</table>

Return Value

The index number that represents the specified index.

Remarks

This is a compile-time function. For more information, see Overview.

Example

The following example returns the index value of the Party index.

static void indexNumExample(Args _arg)
{
    ;

    Global::info(strfmt("%1 is the index number of Party.", indexNum(CustTable, Party)));
}
/**Infolog Display
Message (11:28:03 am)
3 is the index number of Party.
*****/

indexStr

Converts the specified index to a string.

Syntax
**str indexStr(str tableid, str indexid)**

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableid</td>
<td>The table that contains the index.</td>
</tr>
<tr>
<td>indexid</td>
<td>The index to convert.</td>
</tr>
</tbody>
</table>

**Return Value**

A string that represents the specified index.

**Remarks**

This is a compile-time function. For more information, see Overview.

**Example**

The following example assigns the **CashDisc** index value to the *myText* variable.

```cpp
static void fieldStrExample(Args _arg)
{
    str myText;

    myText = fieldStr(CustTable, CashDisc);
    Global::info(strfmt("%1 is the specified index.", myText));
}
/** Infolog Display
Message (09:11:52 am)
CashDisc is the specified index.
****/```

**licenseCodeNum**

Validates that the specified license code exists in the Application Explorer; if not, a compiler error occurs.

**Syntax**

```cpp
int licenseCodeNum(str codename)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>codename</td>
<td>The name of the license code to validate.</td>
</tr>
</tbody>
</table>

**Return Value**

The number of the specified license code.

**Remarks**

This is a compile-time function. For more information, see Overview.

**Example**
static void licenseCodeNumExample(Args args)
{
    int i;
    
    i = licenseCodeNum(SysMorphX);
    Global::info(strfmt("%1 is the license code number for SysMorphX.", i));
}

/****Infolog Display
Message (01:52:35 pm)
24 is the license code number for SysMorphX.
****/

licenseCodeStr
Validates that the specified license code exists in the Application Explorer; if not, a compiler error occurs.

Syntax

str licenseCodeStr(str codename)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>codename</td>
<td>The name of the license code to validate.</td>
</tr>
</tbody>
</table>

Return Value

The name of the specified license code.

Remarks

This is a compile-time function. For more information, see Overview.

Example

static void licenseCodeStrExample(Args _arg)
{
    str s;
    
    s = licenseCodeStr(SysMorphX);
    Global::info(strfmt("%1 is the license code string for SysMorphX.", s));
}

/****Infolog Display
Message (02:33:56 pm)
SysMorphX is the license code string for SysMorphX.
****/

literalStr
Validates that the specified string can be a literal string; if not, a compiler error occurs.

Syntax

str literalStr(int str)

Parameters
###_literalStr

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>codename</td>
<td>The string to validate.</td>
</tr>
</tbody>
</table>

**Return Value**
The literal string if valid.

**Remarks**
This is a compile-time function. For more information, see [Overview](#).

**Example**

```c
{
    str s;
    
    s = literalStr("This is a literal str");
    print s;
    pause;
}
```

### maxDate

Retrieves the maximum value allowed for a variable of type date.

**Syntax**

```c
date maxDate()
```

**Return Value**
The maximum value allowed for a variable of type `date`, which is **2154-12-31**.

**Remarks**
This is a compile-time function. For more information, see [Overview](#).

**Example**

```c
static void maxDateExample(Args _arg)
{
    date maximumDate;
    
    maximumDate = maxDate();
    print maximumDate;
    pause;
}
```

### maxInt

Retrieves the maximum signed value that can be stored in an `int` type.

**Syntax**

```c
int maxInt()
```
The maximum value allowed value of an integer.

**Remarks**
Any other integer will be less than or equal to the returned value. This is a compile-time function. For more information, see [Overview](#).

**Example**
```java
static void maxIntExample(Args _arg)
{
    int i;
    print "The maximum value for type int is " + int2Str(maxInt());
    pause;
}
```

---

### measurementStr

Returns the name of a measurement.

**Syntax**
```java
str measurementStr(str measurement)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>measurement</td>
<td>The name of the measurement.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the measurement.

**Remarks**
This is a compile-time function. For more information, see [Overview](#).

**Example**

No example.

### measureStr

Returns the name of a measure.

**Syntax**
```java
str measureStr(str measure)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>measure</td>
<td>The name of the measure.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the measure.
Remarks
This is a compile-time function. For more information, see Overview.

Example
No example.

menuItemActionStr
Validates that the specified menu item action exists in the Application Object Tree (Application Explorer); if it does not, a compiler error occurs.

Syntax
```plaintext
str menuItemActionStr(class menuItem)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>codename</td>
<td>The name of the menu item action to validate.</td>
</tr>
</tbody>
</table>

Return Value
The name of the menu item action, if it is valid.

Remarks
This is a compile-time function. For more information, see Overview.

Example
```plaintext
{
    str s1, s2, s3, s4;
    
    s1 = menuItemActionStr(AssetCopy);
    s2 = menuItemDisplayStr(Address);
    s3 = menuItemOutputStr(AssetBarcode);
    s4 = menuStr(Administration);

    print "menuItemActionStr for AssetCopy is " + s1;
    print "menuItemDisplayStr for Address is " + s2;
    print "menuItemOutputStr for AssetBarcode is " + s3;
    print "menuStr for Administration is " + s4;

    pause;
}
```

menuItemDisplayStr
Validates that the specified menu item display exists in the Application Explorer; if it does not, a compiler error occurs.

Syntax
```plaintext
str menuItemDisplayStr(class menuItem)
```

Parameters
menuItemActionStr

The name of the menu item display to validate.

Return Value
The name of the specified menu item display, if it is valid.

Remarks
This is a compile-time function. For more information, see Overview.

Example

```c
{
    str s1, s2, s3, s4;
    
    s1 = menuItemActionStr(AssetCopy);
    s2 = menuItemDisplayStr(Address);
    s3 = menuItemOutputStr(AssetBarcode);
    s4 = menuStr(Administration);

    print "menuItemActionStr for AssetCopy is " + s1;
    print "menuItemDisplayStr for Address is " + s2;
    print "menuItemOutputStr for AssetBarcode is " + s3;
    print "menuStr for Administration is " + s4;

    pause;
}
```

menuItemOutputStr

Validates that the specified menu item output exists in the Application Explorer; if not, a compiler error occurs.

Syntax

```c
str menuItemOutputStr(class menuitem)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>codename</td>
<td>The name of the menu item output to validate.</td>
</tr>
</tbody>
</table>

Return Value
The specified menu item output if valid.

Remarks
This is a compile-time function. For more information, see Overview.

Example
menuStr

Validates that the specified menu exists in the Application Explorer; if not, a compiler error occurs.

**Syntax**

```plaintext
str menuStr(class menu)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>menu</td>
<td>The name of the menu to validate.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the specified menu item if valid.

**Remarks**

This is a compile-time function. For more information, see Overview.

**Example**

```plaintext
{
  str s1, s2, s3, s4;
  
  s1 = menuItemActionStr(AssetCopy);
  s2 = menuItemDisplayStr(Address);
  s3 = menuItemOutputStr(AssetBarcode);
  s4 = menuStr(Administration);

  print "menuItemActionStr for AssetCopy is " + s1;
  print "menuItemDisplayStr for Address is " + s2;
  print "menuItemOutputStr for AssetBarcode is " + s3;
  print "menuStr for Administration is " + s4;

  pause;
}
```

methodStr
Validates that the specified method exists in the specified class; if it does not, a compiler error occurs.

**Syntax**

```c
str methodStr(class class, int method)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>The name of the class.</td>
</tr>
<tr>
<td>method</td>
<td>The name of the method to validate.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the specified method, if it is valid.

**Remarks**

This is a compile-time function. For more information, see [Overview](#).

**Example**

```c
{
    #define.timeout(50)
    str s;
    SysHelpInitTimeOut SysHelpInitTimeOut;
    
    s = methodStr(SysHelpInitTimeOut, timeout);
    print s;
    pause;
}
```

**minInt**

Retrieves the minimum signed value that can be stored in an `int` type.

**Syntax**

```c
int minInt()
```

**Return Value**

The minimum value of an `int` type.

**Remarks**

Any other integer value will be greater than or equal to the returned value. This is a compile-time function. For more information, see [Overview](#).

**Example**
static void minIntExample(Args _arg)
{
    int i;
    ;
    i = minInt();
    print "minInt() is " + int2Str(i);
    pause;
}

privilegeStr

Returns the name of the privilege.

Syntax

str privilegeStr(str privilege)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>privilege</td>
<td>The privilege for which to return the name.</td>
</tr>
</tbody>
</table>

Return Value

The name of the privilege.

Remarks

This is a compile-time function. For more information, see Overview.

Example

No example.

queryDatasourceStr

Causes the X++ compiler to check whether the data source exists on the query, and to replace the function call with a string of the valid data source name.

Syntax

str queryDataSourceStr(queryName, dataSourceName)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>queryName</td>
<td>The name of the query, not in quotation marks.</td>
</tr>
<tr>
<td>dataSourceName</td>
<td>The name of the data source that is on the query, not in quotation marks.</td>
</tr>
</tbody>
</table>

Return Value

A string that contains the name of the data source as it appears in the Application Explorer.

Remarks
A compile error is issued if the compiler determines that the data source does not exist on the query. If your X++ code uses a string that contains quotation marks to supply the data source name, the error cannot be discovered until run time. Use of this function enables the compiler to discover the error earlier at compile time. X++ functions such as `queryDataSourceStr` that are executed by the compiler are referred to as compile-time functions or compile-time functions. That is why the input parameters are not standard strings in quotation marks. Compile-time functions are not represented in the p-code or other executable that is output by the compiler. This is a compile-time function. For more information, see Overview.

**Example**

No example.

### queryMethodStr

Returns the name of a method of a query.

**Syntax**

```
str queryMethodStr(str queryName, str methodName)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>queryName</td>
<td>The name of the query.</td>
</tr>
<tr>
<td>methodName</td>
<td>The method of the form.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of a method in a query.

**Remarks**

This is a compile-time function. For more information, see Overview.

**Example**

No example.

### queryStr

Retrieves a string that represents an existing query.

**Syntax**

```
str queryStr(str query)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>The query to retrieve.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the query.

**Remarks**
Example

```csharp
static void queryStrExample(Args _arg)
{
    str myText;
    
    myText = queryStr(AssetTable);
    Global::info(strfmt("%1 is the name of the query.", myText));
}
/****Infolog Display
Message (09:45:16 am)
AssetTable is the name of the query.
****/```

reportStr

Retrieves a string that represents the name of the specified report.

Syntax

```csharp
str reportStr(str report)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>report</td>
<td>The report for which to return the name.</td>
</tr>
</tbody>
</table>

Return Value

The name of the report.

Remarks

This is a compile-time function. For more information, see Overview.

Example

The following example assigns the name of the AssetAddition report to the MyTxt variable.

```csharp
static void reportStrExample(Args _args)
{
    str MyTxt;
    
    MyTxt = reportStr(AssetAddition);
    Global::info(strfmt("%1 is the name of the report.", MyTxt));
}
/****Infolog Display.
Message (10:46:36 am)
AssetAddition is the name of the report.
****/```

resourceStr

Validates that the specified resource exists in the Application Explorer; if it does not, a compiler error occurs.

Syntax
str resourceStr(str resourcename)

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>resourcename</td>
<td>The name of the resource to validate.</td>
</tr>
</tbody>
</table>

**Return Value**
The name of the specified resource, if it is valid.

**Remarks**
This is a compile-time function. For more information, see Overview.

**Example**
```
{
    print "Str for resource StyleSheet_Help_Axapta is "
    + resourceStr(StyleSheet_Help_Axapta);
    pause;
}
```

**roleStr**
Retrieves a string that represents the name of the specified security role.

**Syntax**
```
str roleStr(str securityRole)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>securityRole</td>
<td>The name of the security role.</td>
</tr>
</tbody>
</table>

**Return Value**
The name of the security role in a string.

**Remarks**
This is a compile-time function. For more information, see Overview.

**Example**
No example.

**ssrsReportStr**
Retrieves a string that represents the name of the specified report. Use this function when you want to specify the report that should be run by a report controller class.

**Syntax**
```
str ssrsReportStr(str report, str design)
```
### Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>report</td>
<td>The report to return the name for.</td>
</tr>
<tr>
<td>design</td>
<td>The name of the design that is associated with the report.</td>
</tr>
</tbody>
</table>

#### Return Value

The name of the report.

#### Remarks

The `ssrsReportStr` function parses the two values that are passed to it, to validate whether they belong to a valid report. The report name must be set when a menu item points to a controller(), so that the controller can determine which report-design combination must be invoked. Use of the `ssrsReportStr` function provides the benefit of compile-time validation for the report and design name. This is a compile-time function. For more information, see Overview.

#### Example

```java
public static void main(Args _args)
{
    // Initializing the object for a controller class, in this case, the class named AssetListingController.
    SrssReportRunController controller = new AssetListingController();

    // Getting the properties of the called object (in this case AssetListing MenuItem)
    controller.parmArgs(_args);
    // Setting the Report name for the controller.
    controller.parmReportName(ssrsReportStr(AssetListing, Report));

    // Initiate the report execution.
    controller.startOperation();
}
```

### staticDelegateStr

Returns the name of a static delegate.

#### Syntax

```
str staticDelegateStr(str class, str delegate)
```

#### Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>The name of a class, table, or form.</td>
</tr>
<tr>
<td>delegate</td>
<td>The name of the delegate.</td>
</tr>
</tbody>
</table>

#### Return Value

The name of the delegate.

#### Remarks

This is a compile-time function. For more information, see Overview.
Example
No example.

**staticMethodStr**
Validates that the specified static method exists in the specified class; if it does not, a compiler error occurs.

**Syntax**
```
str staticMethodStr(class class, int method)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>The name of the class.</td>
</tr>
<tr>
<td>method</td>
<td>The name of the static method to validate.</td>
</tr>
</tbody>
</table>

**Return Value**
The name of the static method, if it is valid.

**Remarks**
This is a compile-time function. For more information, see Overview.

**Example**
No example.

**tableCollectionStr**
Validates that the specified table collection exists in the Application Explorer; if it does not, a compiler error occurs.

**Syntax**
```
str tableCollectionStr(class tablecollection)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>tablecollection</td>
<td>The name of the table collection to validate.</td>
</tr>
</tbody>
</table>

**Return Value**
The name of the specified table collection, if it is valid.

**Remarks**
This is a compile-time function. For more information, see Overview.

**Example**
No example.

tableFieldGroupStr
Retrieves the name of a field group as a string.

**Syntax**

```plaintext
str tableFieldGroupStr(str tableName, str fieldGroupName)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableName</td>
<td>The table that contains the field group.</td>
</tr>
<tr>
<td>fieldGroupName</td>
<td>The field group in the table.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the field group as a string.

**Remarks**

This is a compile-time function. For more information, see [Overview](#).

**Example**

The following example retrieves the name of the Editing field group as a string.

```plaintext
static void tableFieldGroupStrExample(Args _arg)
{
    //
    Global::info(tableFieldGroupStr(AccountingDistribution, Editing));
}  
/****Infolog Display
Message (03:14:54 pm)
Editing
****/
```

---

**tableMethodStr**

Validates that the specified method exists in the specified table; if it does not, a compiler error occurs.

**Syntax**

```plaintext
str tableMethodStr(int table, int method)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>The name of the table.</td>
</tr>
<tr>
<td>method</td>
<td>The name of the method to validate.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the method, if it is valid.

**Remarks**

This is a compile-time function. For more information, see [Overview](#).
Example
No example.

tableNum
Retrieves the table ID of the specified table.

Syntax

```plaintext
int tableNum(str table)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>The table to retrieve the table ID for</td>
</tr>
</tbody>
</table>

Return Value
The table ID of the specified table.

Remarks
This is a compile-time function. For more information, see Overview.

Example
The following example sets the `tableID` variable to 77, which is the ID of the `CustTable` table.

```plaintext
static void tableNumExample(Args _args)
{
    int tableID;
    ;

    tableID = tableNum(CustTable);
    Global::info(strfmt("%1 is the table ID for the CustTable table.", tableID));

}
/****Infolog Display
Message (11:15:54 am)
77 is the table ID for the CustTable table.
****/
```

tablePName
Retrieves a string that contains the printable name of the specified table.

Syntax

```plaintext
str tablePName(str table)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>The table to retrieve the printable name for</td>
</tr>
</tbody>
</table>

Return Value
The name of the specified table.

Remarks
This is a compile-time function. For more information, see Overview.

Example
The following example assigns the label of the **CustTable** table to the *MyText* variable.

```java
static void tablePNameExample(Args _args)
{
    str MyText;
    MyText = tablePname(CustTable);
    Global::info(strfmt("%1 is the label of the CustTable table.", MyText));
}
/**** Infolog Display.
Message (12:13:53 pm)
Customers is the label of the CustTable table.
****/
```

tableStaticMethodStr
Validates that the specified static method exists in the specified table; if it does not, a compiler error occurs.

Syntax

```java
str tableStaticMethodStr(int table, int method)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>The name of the table.</td>
</tr>
<tr>
<td>method</td>
<td>The name of the static method to validate.</td>
</tr>
</tbody>
</table>

Return Value
The name of the specified static method.

Remarks
This is a compile-time function. For more information, see Overview.

Example
No example.

tableStr
Retrieves a string that contains the name the specified table.

Syntax

```java
str tableStr(str table)
```

Parameters
### table

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>The table to retrieve a string for.</td>
</tr>
</tbody>
</table>

#### Return Value

A string value that contains the name of the specified table.

#### Remarks

This is a compile-time function. For more information, see Overview.

#### Example

The following example assigns the name of the **CustTable** table to the `MyTxt` variable.

```cpp
static void tableStrExample(Args _args)
{
    str MyTxt;
    
    MyTxt = tableStr(CustTable);
    Global::info(strfmt("%1 is the str output of the input of CustTable.", MyTxt));
}
/*** InfoLog Display.
Message (01:21:49 pm)
CustTable is the str output of the input of CustTable.
****/```

### tileStr

Retrieves a string that represents the name of the specified tile.

#### Syntax

`str tileStr(str tile)`

#### Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>tile</td>
<td>The name of the tile.</td>
</tr>
</tbody>
</table>

#### Return Value

The name of the tile in a string.

#### Remarks

This is a compile-time function. For more information, see Overview.

#### Example

No example.

### varStr

Retrieves a string that contains the name of the specified variable.

#### Syntax
str varStr(str var)

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>The name of the variable.</td>
</tr>
</tbody>
</table>

**Return Value**

A string that contains the name of the specified variable.

**Remarks**

This is a compile-time function. For more information, see [Overview](#).

**Example**

```java
static void varStrExample(Args _arg)
{
    str myString;
    anytype myVariable;
    
    myString = varStr(myVariable);
    Global::info(strfmt("%1 is the variable name.", myString));
}
/****Infolog Display.
Message (02:26:56 pm)
myVariable is the variable name.
****/
```

**webActionItemStr**

Validates that the specified web action item exists in the Application Explorer; if it does not, a compiler error occurs.

**Syntax**

```java
str webActionItemStr(class webactionitem)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>webactionitem</td>
<td>The name of the web action item to validate.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the specified web action item, if it is valid.

**Remarks**

This is a compile-time function. For more information, see [Overview](#).

**Example**

```java
```
```csharp
str s;
;
s = webActionItemStr(EPFlushData);
print "webactionitem str is " + s;
pause;
}
```

**webDisplayContentItemStr**

Validates that the specified web display content item exists in the Application Explorer; if it does not, a compiler error occurs.

**Syntax**

```csharp
str webDisplayContentItemStr(class webdisplaycontentitem)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>webdisplaycontentitem</td>
<td>The name of the web display content item to validate.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the specified web display content item, if it is valid.

**Remarks**

This is a compile-time function. For more information, see **Overview**.

**Example**

```csharp
{
    str s;
    
    s = webDisplayContentItemStr(EPAdmin);
    print "string for webcontent display item EPAdmin is " + s;
    pause;
}
```

**webFormStr**

Validates that the specified web form exists in the Application Explorer; if it does not, a compiler error occurs.

**Syntax**

```csharp
str webFormStr(str name)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the web form to validate.</td>
</tr>
</tbody>
</table>

**Return Value**
The name of the specified web form, if it is valid.

**Remarks**
This is a compile-time function. For more information, see Overview.

**Example**
```perl
{
    str s;
    s = webFormStr(EPAdmin);
    print "String for web form EPAdmin is " + s;
    pause;
}
```

---

**webletItemStr**

Validates that the specified weblet item exists in the Application Explorer; if it does not, a compiler error occurs.

**Syntax**
```perl
str webletItemStr(class webletitem)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>webletitem</td>
<td>The name of the weblet item to validate.</td>
</tr>
</tbody>
</table>

**Return Value**
The name of the specified weblet item, if it is valid.

**Remarks**
This is a compile-time function. For more information, see Overview.

**Example**
```perl
{
    str s;
    s = webletItemStr(WebFormWeblet);
    print "String for WebFormWeblet is " + s;
    pause;
}
```

---

**webMenuStr**

Validates that the specified web menu exists in the Application Explorer; if it does not, a compiler error occurs.

**Syntax**
```perl
str webMenuStr(str name)
```

**Parameters**
webMenuStr

Validates that the specified web menu exists in the Application Explorer; if it does not, a compiler error occurs.

**Syntax**

```java
str webMenuStr(webMenu)  
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>webMenu</td>
<td>The name of the web menu to validate.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the specified web menu, if it is valid.

**Remarks**

This is a compile-time function. For more information, see Overview.

**Example**

```java
{  
  str s;  
  s = webMenuStr(ECPAdmin);  
  print "String for web menu ECPAdmin is " + s;  
  pause;  
}
```

---

webOutputContentItemStr

Validates that the specified web output content item exists in the Application Explorer; if it does not, a compiler error occurs.

**Syntax**

```java
str webOutputContentItemStr(weboutputcontentitem)  
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>weboutputcontentitem</td>
<td>The name of the web output content item to validate.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the specified web output content item, if it is valid.

**Remarks**

This is a compile-time function. For more information, see Overview.

**Example**

```java
{  
  str s;  
  s = webOutputContentItemStr(EPPriceList);  
  print "string for weboutput content item EPPriceList is " + s;  
  pause;  
}
```

---

webpageDefStr

Validates that the specified Web page definition exists in the Application Explorer; if it does not, a compiler error occurs.
**Syntax**

```java
str webpageDefStr(str pagename)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>pagename</td>
<td>The name of the Web page definition to validate.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the specified web-page definition, if it is valid.

**Remarks**

This is a compile-time function. For more information, see Overview.

**Example**

No example.

**webReportStr**

Validates that the specified web report exists in the Application Explorer; if it does not, a compiler error occurs.

**Syntax**

```java
str webReportStr(str name)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the web report to validate.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the specified web report, if it is valid.

**Remarks**

This is a compile-time function. For more information, see Overview.

**Example**

```java
{
    str s;
    s = webReportStr(EPCSSalesConfirm);
    print "String for web report EPCSSalesConfirm is " + s;
    pause;
}
```

**websiteDefStr**

Validates that the specified web-site definition exists in the Application Explorer; if it does not, a compiler error occurs.

**Syntax**
str websiteDefStr(str resourcename)

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>resourcename</td>
<td>The name of the Web site definition to validate.</td>
</tr>
</tbody>
</table>

**Return Value**
The name of the specified web-site definition, if it is valid.

**Remarks**
This is a compile-time function. For more information, see Overview.

**Example**

```java
{
    str s;
    
    s = websiteDefStr(AxSiteDef_1033_xip);
    print "string for web site definition AxSiteDef_1033_xip is " + s;
    pause;
}
```

**webSiteTempStr**

Validates that the specified web-site template exists in the Application Explorer; if it does not, a compiler error occurs.

**Syntax**

```java
str websiteTempStr(str resourcename)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>resourcename</td>
<td>The name of the Web site template to validate.</td>
</tr>
</tbody>
</table>

**Return Value**
The name of the specified web-site template, if it is valid.

**Remarks**
This is a compile-time function. For more information, see Overview.

**Example**
No example.

**webStaticFileStr**

Validates that the specified web static file exists in the Application Explorer; if it does not, a compiler error occurs.

**Syntax**
webStaticFileStr

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>pagename</td>
<td>The name of the web static file to validate.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the specified web static file, if it is valid.

**Remarks**

This is a compile-time function. For more information, see Overview.

**Example**

```plaintext
{
    str s;
    
    s = webStaticFileStr(AXEP);
    print "string for web static file AXEP is " + s;
    pause;
}
```

webUrlItemStr

Validates that the specified web URL item exists in the Application Explorer; if it does not, a compiler error occurs.

**Syntax**

```plaintext
str webUrlItemStr(class weburlitem)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>weburlitem</td>
<td>The name of the web URL item to validate.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the specified web URL item, if it is valid.

**Remarks**

This is a compile-time function. For more information, see Overview.

**Example**
{ str s;
   s = webUrlItemStr(EPAdmin);
   print "string for web url item EPAdmin is " + s;
   pause;
}

webWebPartStr
Validates that the specified web part exists in the Application Explorer; if it does not, a compiler error occurs.

Syntax

str webWebpartStr(str resourcename)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>resourcename</td>
<td>The name of the web part to validate.</td>
</tr>
</tbody>
</table>

Return Value
The name of the specified web part, if it is valid.

Remarks
This is a compile-time function. For more information, see Overview.

Example

{ str s;
   s = webWebpartStr(AxWebParts_cab);
   print "string for web part AxWebParts_cab is " + s;
   pause;
}

workflowApprovalStr
Retrieves the name of a workflow approval in the Application Object Tree (Application Explorer) as a string.

Syntax

str workflowapprovalstr(approval approval)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>approval</td>
<td>The Application Explorer name of the workflow approval.</td>
</tr>
</tbody>
</table>

Return Value
A string that represents the Application Explorer name of the workflow approval.

Remarks
Use this function instead of literal text to retrieve a string that contains the workflow approval name. If the workflow approval does not exist, the function generates a syntax error at compile time. This is a compile-time function. For more information, see Overview.

Example
The following code example sets the variable str s to MyWorkflowApproval which is the name of the workflow approval in the Application Explorer.

```c
static void MyWorkflowApprovalStrExample(Args _args)
{
    str s;
    s = workflowapprovalstr(MyWorkflowApproval);
    print s;
    pause;
}
```

workflowCategoryStr
Retrieves the name of a workflow category in the Application Object Tree (Application Explorer) as a string.

Syntax

```
str workflowcategorystr(category category)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>category</td>
<td>The Application Explorer name of the workflow category.</td>
</tr>
</tbody>
</table>

Return Value
A string that represents the Application Explorer name of the workflow category.

Remarks
Use this function instead of literal text to retrieve a string that contains the workflow category name. If the workflow category does not exist, the function generates a syntax error at compile time. This is a compile-time function. For more information, see Overview.

Example
The following code example sets the variable str s to MyWorkflowCategory which is the name of the workflow category in the Application Explorer.

```c
static void MyWorkflowCategoryStrExample(Args _args)
{
    str s;
    s = workflowcategorystr(MyWorkflowCategory);
    print s;
    pause;
}
```
**workflowTaskStr**

Retrieves the name of a workflow task in the Application Object Tree (Application Explorer) as a string.

**Syntax**

```plaintext
str workflowtaskstr(task task)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>task</td>
<td>The Application Explorer name of the workflow task.</td>
</tr>
</tbody>
</table>

**Return Value**

A string that represents the Application Explorer name of the workflow task.

**Remarks**

Use this function instead of literal text to retrieve a string that contains the workflow task name. If the workflow task does not exist, the function generates a syntax error at compile time. This is a compile-time function. For more information, see [Overview](#).

**Example**

The following code example sets the variable `str s` to `MyWorkflowTask` which is the name of the workflow task in the Application Explorer.

```plaintext
static void MyWorkflowTaskStrExample(Args _args)
{
    str s;
    s = workflowtaskstr(MyWorkflowTask);
    print s;
    pause;
}
```

**workflowTypeStr**

Validates that the specified workflow type exists in the Application Explorer; if it does not, a compiler error occurs.

**Syntax**

```plaintext
str workflowTypeStr(str workflow)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflow</td>
<td>The name of the workflow type to validate.</td>
</tr>
</tbody>
</table>

**Return Value**

The name of the workflow type.

**Remarks**

...
This is a compile-time function. For more information, see Overview.

Example

```c
static void workFlowTypeStrExample(Args _args)
{
    str s;
    s = workFlowTypeStr(BudgetAccountEntryType);
    print s;
    pause;
}
```
This topic describes the X++ run-time functions.

The X++ language provides nearly 200 system functions that aren't part of any class and are executed at run time. Run-time functions are used for data type conversions, mathematical operations, and so on. Here are some common run-time functions:

- **str2Int** – Creates an int value from a str value.
- **abs** – Creates a positive real value from a real value that is either positive or negative.
- **conFind** – Retrieves the location of an element in a container.

### Call run-time functions from .NET

The logic of the X++ run-time functions is also implemented in the following .NET assembly.

```plaintext
Microsoft.Dynamics.AX.Xpp.Support.DLL
```

Inside this assembly, the X++ run-time functions are implemented as static methods of the following class.

```plaintext
Microsoft.Dynamics.AX.Xpp.PredefinedFunctions
```

### Categories and functions

The following table lists and describes only the categories of X++ functions. These categories are intended to help you understand the many functions. However, the categories don’t represent any formal construct.

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business</td>
<td>Functions that enter financial data and calculate formulas. For more information, see X++ Business Run-Time Functions.</td>
</tr>
<tr>
<td>Container</td>
<td>Functions that operate on the container data type of X++. For more information, see X++ Container Run-Time Functions.</td>
</tr>
<tr>
<td>Conversion</td>
<td>Functions that translate data of one type into data of another type. For more information, see X++ Conversion Run-Time Functions.</td>
</tr>
<tr>
<td>Date</td>
<td>Functions that operate on the date data type. For more information, see X++ Date Run-Time Functions.</td>
</tr>
<tr>
<td>Math</td>
<td>Functions that perform mathematical calculations. For more information, see X++ Math Run-Time Functions.</td>
</tr>
<tr>
<td>CATEGORY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Reflection</td>
<td>Functions that access the metadata about objects and return other metadata about them. For more information, see X++ Reflection Run-Time Functions.</td>
</tr>
<tr>
<td>Session</td>
<td>Functions that change or report on the context of the current user connection. For more information, see X++ Session Run-Time Functions.</td>
</tr>
<tr>
<td>String</td>
<td>Functions that operate on the str data type. For more information, see X++ String Run-Time Functions.</td>
</tr>
<tr>
<td>Other</td>
<td>beep, newGuid, sleep</td>
</tr>
</tbody>
</table>

### Business

For more information, see X++ Business Run-Time Functions.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>cTerm</td>
<td>ddb</td>
<td>dg</td>
<td>fV</td>
</tr>
<tr>
<td>idg</td>
<td>intvMax</td>
<td>intvName</td>
<td>intvNo</td>
</tr>
<tr>
<td>intvNorm</td>
<td>pmt</td>
<td>pt</td>
<td>pv</td>
</tr>
<tr>
<td>rate</td>
<td>sln</td>
<td>syd</td>
<td>term</td>
</tr>
</tbody>
</table>

### Container

For more information, see X++ Container Run-Time Functions.

- conDel
- conFind
- conIns
- conLen
- conNull
- conPeek
- conPoke

### Conversion

For more information, see X++ Conversion Run-Time Functions.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>any2Date</td>
<td>any2Enum</td>
<td>any2Guid</td>
<td>any2Int</td>
</tr>
<tr>
<td>any2Int64</td>
<td>any2Real</td>
<td>any2Str</td>
<td>anytodate</td>
</tr>
<tr>
<td>anytoenum</td>
<td>anytoguid</td>
<td>anytoint</td>
<td>anytoint64</td>
</tr>
</tbody>
</table>
### Date

For more information, see [X++ Date Run-Time Functions.](X++ Date Run-Time Functions)

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>dayName</td>
<td>dayOfMth</td>
<td>dayOfWk</td>
<td>dayOfYr</td>
</tr>
<tr>
<td>endMth</td>
<td>mkDate</td>
<td>mthName</td>
<td>mthOfYear</td>
</tr>
<tr>
<td>nextMth</td>
<td>nextQtr</td>
<td>nextYr</td>
<td>prevMth</td>
</tr>
<tr>
<td>prevQtr</td>
<td>prevYr</td>
<td>systemDateGet</td>
<td>systemDateSet</td>
</tr>
<tr>
<td>timeNow</td>
<td>today</td>
<td>wkOfYear</td>
<td>year</td>
</tr>
</tbody>
</table>

### Math

For more information, see [X++ Math Run-Time Functions.](X++ Math Run-Time Functions)

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs</td>
<td>acos</td>
<td>asin</td>
<td>atan</td>
</tr>
<tr>
<td>corrFlagGet</td>
<td>corrFlagSet</td>
<td>cos</td>
<td>cosh</td>
</tr>
<tr>
<td>decRound</td>
<td>exp</td>
<td>exp10</td>
<td>frac</td>
</tr>
<tr>
<td>log10</td>
<td>logN</td>
<td>max</td>
<td>min</td>
</tr>
<tr>
<td>power</td>
<td>round</td>
<td>sin</td>
<td>sinh</td>
</tr>
<tr>
<td>tan</td>
<td>tanh</td>
<td>trunc</td>
<td></td>
</tr>
</tbody>
</table>

### Reflection

For more information, see [X++ Reflection Run-Time Functions.](X++ Reflection Run-Time Functions)
classIdGet | dimOf | fieldId2Name | fieldId2PName |
fieldId2Name | indexId2Name | indexName2Id | refPrintAll |
tableId2Name | tableId2PName | tableName2Id | typeOf |

**Session**

For more information, see *X++ Session Run-Time Functions*.

curExt | curUserId | funcName | getCurrentPartition |
geGetCurrentPartitionRecId | getPrefix | sessionId | prmIsDefault |
runAs | setPrefix |

**String**

For more information, see *X++ String Run-Time Functions*.

match | strAlpha | strCmp | strColSeq |
strDel | strFind | strFmt | strIns |
strKeep | strLen | strLine | strLTrim |
strUpr | strNFind | strPoke | strPrompt |
strRem | strRep | strRTrim | strScan |

**beep**

Emits a short sound from the speakers on the computer.

```java
void beep()
```

**beep example**

```java
static void beepExample(Args _args)
{
    beep();
}
```
newGuid

Creates a globally unique identifier (GUID).

```c
newGuid()  
```

**Return value**

A GUID.

**newGuid example**

The following example creates a GUID.

```c
static void newGuidExample(Args _arg)
{
    guid myGuid;

    myGuid = newguid();
    print strfmt("The GUID is: %1", myGuid);
}
```

sleep

Pauses the execution of the current thread for the specified number of milliseconds.

```c
sleep(int _duration)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_duration</td>
<td>The number of milliseconds to pause.</td>
</tr>
</tbody>
</table>

**sleep return value**

The number of milliseconds that the thread actually paused.

**Example**

```c
static void sleepExample(Args _arg)
{
    int seconds = 10;
    int i;

    i = sleep(seconds*1000);
    print "job slept for " + int2str(i/1000) + " seconds";
}
```
This topic describes the business run-time functions.

These functions enter financial data and calculate formulas.

cTerm
Calculates the number of periods that are required for the current investment value to yield a target value.

**Syntax**

```plaintext
real cTerm(real interest, real future_value, real current_value)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>interest</td>
<td>The interest rate.</td>
</tr>
<tr>
<td>future_value</td>
<td>The target value.</td>
</tr>
<tr>
<td>current_value</td>
<td>The current investment value.</td>
</tr>
</tbody>
</table>

**Return value**
The number of periods that are required in order to reach `future_value`.

**Remarks**
The `current_value` and `future_value` parameters must have the same prefixed sign (plus or minus).

**Example**

```plaintext
static void cTermExample(Args _arg)
{
    real r;
    r = cTerm(10.0, 500.00, 100.00);
    print "The cTerm is " + num2Str(r, 2, 2, 1, 1);
    pause;
}
```

ddb
Calculates the accelerated depreciation of an asset.

**Syntax**

```plaintext
real ddb(real price, real scrap, real life, int period)
```

**Parameters**
### Return value

The depreciation of the asset.

### Remarks

The book value for a specific period is equal to the purchase price minus the accumulated depreciation for previous periods:

- Book value for Period 1 = Price
- Book value for Period 2 = Book value for Period 1 – Depreciation for Period 1
- Book value for Period n = Book value for Period (n–1) – Depreciation for Period (n–1)

There are three variations for the calculation of depreciation: If Period > Life:

- Depreciation = 0

If (Book value for Period n) – ((Book value for Period n) × 2 ÷ Life) < Residual value:

- Depreciation = (Book value for Period n) – Residual value

In all other cases: Depreciation = (Book value for Period n) × 2 ÷ Life

The `syd` and `sln` functions also calculate the depreciation of an asset. The `syd` and `ddb` functions enables higher depreciation for the earlier years, whereas `sln` calculates a linear depreciation.

```plaintext
ddb(12000,2000,10,1); //Returns the value 2400.
ddb(12000,2000,10,3); //Returns the value 1536.
```

### dg

Calculates the contribution ratio, which is based on the sales price and the purchase price. If the value of the `sale` parameter is 0.0, the calculation can’t be done.

#### Syntax

```plaintext
real dg(real sale, real purchase)
```

#### Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>sale</td>
<td>The sale price.</td>
</tr>
<tr>
<td>purchase</td>
<td>The purchase price.</td>
</tr>
</tbody>
</table>

#### Return value
The contribution ratio.

Remarks

\[
dg(1000, 300); \quad \text{//Returns the value 0.7.} \\
dg(100, 30); \quad \text{//Returns the value 0.7.} \\
dg(20000, 11000); \quad \text{//Returns the value 0.45.}
\]

\[fV\]
Calculates the future value of an investment.

Syntax

\[
\text{real } fV(\text{real amount, real interest, real life})
\]

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>amount</td>
<td>The amount that was paid in during each period.</td>
</tr>
<tr>
<td>interest</td>
<td>The interest rate.</td>
</tr>
<tr>
<td>life</td>
<td>The number of investment periods.</td>
</tr>
</tbody>
</table>

Return value

The future value of the investment.

Remarks

\[
fV(100, 0.14, 10); \quad \text{//Returns the value 1933.73.} \\
fV(400, 0.10, 5); \quad \text{//Returns the value 2442.04.}
\]

\[idg\]
Calculates the sale price, based on the purchase price and the contribution ratio.

\[
\text{real } idg(\text{real purchase, real contribution_ratio})
\]

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>purchase</td>
<td>The purchase price.</td>
</tr>
<tr>
<td>contribution_ratio</td>
<td>The contribution ratio.</td>
</tr>
</tbody>
</table>

Return value

The sale price.

Remarks

If the contribution ratio is equal to 1.0, the calculation can't be done. The \text{idg} function is the inverse of the \text{dg}
idg(300,0.7); //Returns the value 1000.
idg(11000,0.45); //Returns the value 20000.

intvMax

Retrieves the number of intervals for the specified period when the period is divided into parts as specified by the *func* parameter.

```plaintext
int intvMax(date input_date, date ref_date, int func)
```

### Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>input_date</td>
<td>The end of the period, which must be later than the <em>ref_date</em> parameter.</td>
</tr>
<tr>
<td>ref_date</td>
<td>The start of the period.</td>
</tr>
<tr>
<td>func</td>
<td>A <em>IntvScale</em> system enumeration value that indicates the division unit.</td>
</tr>
</tbody>
</table>

### Remarks

Here are the possible values for the *func* parameter:

- None
- YearMonthDay
- YearMonth
- Year
- MonthDay
- Month
- Day
- YearQuarter
- Quarter
- YearWeek
- Week
- WeekDay

### Example

```plaintext
static void intvMaxExample()
{
    date refDate = str2Date("4/9/2007", 213);
    date inputDate = str2Date("10/5/2007", 213);
    int numberOfIntervals;
    numberOfIntervals = intvMax(inputDate, refDate, intvScale::YearMonth);
    print numberOfIntervals;
    pause;
}
```
intvName

Returns the name of the interval that is the specified number of intervals ahead of the specified date.

\[
\text{str intvName(date input_date, int col, enum func)}
\]

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>input_date</td>
<td>A date in the first interval.</td>
</tr>
<tr>
<td>col</td>
<td>The number of intervals ahead of the date that is specified by the input_date parameter.</td>
</tr>
<tr>
<td>func</td>
<td>An intvScale enumeration value.</td>
</tr>
</tbody>
</table>

**Return value**

The name of the interval.

**Remarks**

For example, if the func parameter is the IntvScale::WeekDay enumeration value, this method returns the name of the weekday. If the func parameter is the IntvScale::Week enumeration value, this method returns a string that contains the number of the week.

**Example**

```c
static void intvNameExample(Args _args)
{
    date refDate = 2672010;
    str name;
    name = intvName(refDate, 3, IntvScale::WeekDay);
    Global::info(strfmt("%1 is the output, which indicates the day of the week 3 days after 26\7\2010.", name));
}
/**** Infolog display.
Message (09:56:55 am)
Thu is the output, which indicates the day of the week 3 days after 2672010.
****/
```

intvNo

Calculates the number of intervals between two dates when you divide the time into the specified intervals.

**Syntax**

\[
\text{int intvNo(date input_date, date ref_date, int func)}
\]

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>input_date</td>
<td>A date that indicates the end of the period</td>
</tr>
</tbody>
</table>
**PARAMETER** | **DESCRIPTION**
---|---
**ref_date** | A date that indicates the start of the period.
**func** | An **intvScale** enumeration value.

### Return value
The number of intervals between the dates that are specified by the **ref_date** and **input_date** parameters.

### Example
```c
code
static void intvNoExample(Args _args)
{
  date inputDate = str2Date("1/1/2007", 213);
  date refDate = str2Date("3/1/2007", 213);
  int noOfIntervals;
  noOfIntervals = intvNo(refDate, inputDate, intvScale::Month);
  print noOfIntervals;
  pause;
  //noOfIntervals now holds the difference in months between March and January (2).
}
```

### intvNorm
Returns the normalized date for the period.

#### Syntax
```c
code
date intvNorm(date input_date, date ref_date, int func)
```

#### Parameters

<table>
<thead>
<tr>
<th><strong>PARAMETER</strong></th>
<th><strong>DESCRIPTION</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>input_date</strong></td>
<td>The end of the period, which must be later than the date that is specified by the <strong>ref_date</strong> parameter.</td>
</tr>
<tr>
<td><strong>ref_date</strong></td>
<td>The start of the period.</td>
</tr>
<tr>
<td><strong>func</strong></td>
<td>An <strong>intvScale</strong> enumeration value that indicates the interval division unit.</td>
</tr>
</tbody>
</table>

#### Return value
The normalized date for the period.

#### Remarks
The returned date will equal the date of the first day in the interval in which the date that is specified by the **ref_date** parameter exists.

#### Example
```java
static void example()
{
    print intvNorm(today(), today()-1, IntVScale::WeekDay);
    pause;
}
```

### pmt

Calculates the amount that must be paid every period to repay a loan.

**Syntax**

```java
real pmt(real principal, real interest, real life)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>principal</td>
<td>The amount that was originally borrowed.</td>
</tr>
<tr>
<td>interest</td>
<td>The interest that is applied each period to the amount that was borrowed.</td>
</tr>
<tr>
<td>life</td>
<td>The number of periods that the loan is repaid over</td>
</tr>
</tbody>
</table>

**Return value**

The amount that must be paid every period.

**Remarks**

The `life` and `interest` parameters must be expressed in the same time units. The value of the `life` parameter must be more than 0.0.

**Example**

```java
pmt(4000,0.14,4); //Returns the value 1372.82.
pmt(10000,0.10,20); //Returns the value 1174.60.
```

### pt

Retrieves the sum of a number plus a specified percentage of that number.

**Syntax**

```java
real pt(real amount, real percentage)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>amount</td>
<td>The original number.</td>
</tr>
<tr>
<td>percentage</td>
<td>The percentage supplement.</td>
</tr>
</tbody>
</table>
Return value
The number that is equal to \((\text{amount} \times \text{percentage}) + \text{amount}\).

Remarks

\[
\text{pt}(2000.0,0.10); \quad //\text{Returns the value 2200.0.}
\]
\[
\text{pt}(20.0,0.10); \quad //\text{Returns the value 22.0.}
\]

pv
Calculates the present value of an annuity, where an amount is received over multiple periods and the interest rate is deducted for each period.

Syntax

\[
\text{real pv(real amount, real interest, real life)}
\]

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>amount</td>
<td>The amount that is paid during each period.</td>
</tr>
<tr>
<td>interest</td>
<td>The interest rate.</td>
</tr>
<tr>
<td>life</td>
<td>The number of times that the value that is specified by the amount parameter is paid.</td>
</tr>
</tbody>
</table>

Return value
The current value of an annuity.

Remarks

\[
\text{pv(300,0.14,4);} \quad //\text{Returns the value 874.11.}
\]

rate
Calculates the interest that is required for the current investment value to attain the future value over the specified number of periods.

Syntax

\[
\text{real rate(real _future_value, real _current_value, real _terms)}
\]

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_future_value</td>
<td>The future value of the investment.</td>
</tr>
<tr>
<td>_current_value</td>
<td>The current value of the investment.</td>
</tr>
<tr>
<td>_terms</td>
<td>The number of periods that the investment spans.</td>
</tr>
</tbody>
</table>
Return value
The calculated interest rate.

Remarks

rate(10000,1000,20); //Returns the value 0.12.

sln
Retrieves the constant depreciation amount for the specified asset for each depreciation period.

Syntax

real sln(real price, real scrap, real life)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>price</td>
<td>The purchase price of the asset.</td>
</tr>
<tr>
<td>scrap</td>
<td>The scrap value of the asset.</td>
</tr>
<tr>
<td>life</td>
<td>The number of periods in the expected life of the asset.</td>
</tr>
</tbody>
</table>

Return value
The depreciation amount.

Example

static void slnExample(Args _arg)
{
    real r;
    r = sln(100.00, 50.00, 50.00);
    print r;
    pause;
}

syd
Calculates the depreciation of an asset over a specified period.

Syntax

real syd(real _price, real _scrap, real _life, int _period)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_price</td>
<td>The purchase price of the asset.</td>
</tr>
<tr>
<td>_scrap</td>
<td>The scrap value of the asset.</td>
</tr>
<tr>
<td>PARAMETER</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>_life</td>
<td>The expected life of the asset (the number of periods).</td>
</tr>
<tr>
<td>_period</td>
<td>The period to calculate depreciation for.</td>
</tr>
</tbody>
</table>

**Return value**
The amount of depreciation over the specified period.

**Remarks**
In contrast to the **sln** function, the **syd** function can allow for an accelerated depreciation of the asset. As with the **ddb** function, this enables higher depreciation during the early periods of the life of an asset.

**Example**
In the following examples, the periodic depreciation is calculated for an asset that has a purchase price of 10,000, a scrap value of 2,000, and a life of 5. In comparison, **sln(10000,2000,5)** would calculate 1600.00 for each period.

```plaintext
// Returns the value 2666.67 (for the 1st period).
syd(10000,2000,5,1);
// Returns the value 2133.33 (for the 2nd period).
syd(10000,2000,5,2);
// Returns the value 1600.00 (for the 3rd period).
syd(10000,2000,5,3);
// Returns the value 1066.67 (for the 4th period).
syd(10000,2000,5,4);
// Returns the value 533.33 (for 5th - and final- period).
syd(10000,2000,5,5);
```

term
Calculates the number of periods that an investment must run for.

**Syntax**

```plaintext
real term(real amount, real interest, real future_value)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>amount</td>
<td>The amount of the periodic investment.</td>
</tr>
<tr>
<td>interest</td>
<td>The interest rate for each period.</td>
</tr>
<tr>
<td>future_value</td>
<td>The future value that is anticipated for the investment</td>
</tr>
</tbody>
</table>

**Return value**
The number of periods that the investment must run for.

**Example**
static void termExample(Args _args)
{
    print term(400, 0.08, 5000);  //returns the value '9.01'.
    print term(100, 0.14, 3000);  //returns the value '12.58'.
    pause;
}
This topic describes the container run-time functions. These functions manipulate the contents of containers.

**conDel**

Removes the specified number of elements from a container.

**Syntax**

```cpp
container conDel(container container, int start, int number)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>The container to remove elements from.</td>
</tr>
<tr>
<td>start</td>
<td>The one-based position at which to start removing elements.</td>
</tr>
<tr>
<td>number</td>
<td>The number of elements to delete.</td>
</tr>
</tbody>
</table>

**Return value**

A new container that doesn't include the removed elements.

**Example**

```cpp
static void conDelExample(Args _args) {
    container c = ["Hello world", 1, 3.14];
    // Deletes the first two items from the container.
    c = conDel(c, 1, 2);
}
```

**conFind**

Locates the first occurrence of an element or a sequence of elements in a container.

**Syntax**

```cpp
int conFind (container container, anytype element, ...)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>The container to search.</td>
</tr>
</tbody>
</table>
### Remarks

If several elements are specified in the sequence, they must be separated by commas and specified in the correct sequence. The elements can be of any data type.

### Return value

0 if the item was not found; otherwise, the sequence number of the item.

### Example

```csharp
static void conFindExample(Args _args)
{
    container c = ["item1", "item2", "item3"];
    int i;
    int j;
    i = conFind(c, "item2");
    j = conFind(c, "item4");
    print "Position of 'item2' in container is " + int2Str(i);
    print "Position of 'item4' in container is " + int2Str(j);
}
```

### conIns

Inserts one or more elements into a container.

#### Syntax

```csharp
container conIns (container container, int start, anytype element, ... )
```

#### Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>The container to insert elements into.</td>
</tr>
<tr>
<td>start</td>
<td>The position to insert elements at.</td>
</tr>
<tr>
<td>element</td>
<td>One or more elements to insert, separated by commas.</td>
</tr>
</tbody>
</table>

#### Return value

A new container that contains the inserted elements.

#### Remarks

The first element of the container is specified by the number 1. To insert after the n element, the `start` parameter should be n+1. You can also use the `+=` operator to add values of any type to a container. For example, to create a container that contains the squared values of the first 10 loop iterations, use the following code.
```c
int i;
container c;

for (i = 1; i <= 10; i++)
{
    c += i*i;
}
```

**Example**

```c
static void conInsExample(Args _arg)
{
    container c;
    int i;

    c = conIns(c,1,"item1");
    c = conIns(c,2,"item2");
    for (i = 1 ; i <= conLen(c) ; i++)
    {
        // Prints the content of a container.
        print conPeek(c, i);
    }
}
```

### conLen

Retrieves the number of elements in a container.

**Syntax**

```c
int conLen(container container)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>The container to count the number of elements in.</td>
</tr>
</tbody>
</table>

**Return value**

The number of elements in the container.

**Example**

```c
static void conLenExample(Args _arg)
{
    container c;
    int i;

    c = conIns(["item1", "item2"], 1);
    for (i = 1 ; i <= conLen(c) ; i++)
    {
        print conPeek(c, i);
    }
}
```

### conNull
container conNull()

**Remarks**

Use this function to explicitly dispose of the contents of a container.

**Return value**

An empty container.

**Example**

```c
static void conNullExample(Args _arg) {
  container c = ["item1", "item2", "item3"];  

  print "Size of container is " + int2str(conLen(c));  
  // Set the container to null.  
  c = conNull();  
  print "Size of container after conNull() is " + int2str(conLen(c));
}
```

**conPeek**

Retrieves a specific element from a container and converts it into another data type, if conversion is required.

**Syntax**

```c
anytype conPeek(container container, int number)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>The container to return an element from.</td>
</tr>
<tr>
<td>number</td>
<td>The position of the element to return. Specify 1 to get the first element. An invalid position number, such as -3, 0, or a number that is higher than the length of the container, might cause unpredictable errors.</td>
</tr>
</tbody>
</table>

**Return value**

The element in the container at the position that is specified by the *number* parameter. The `conPeek` function automatically converts the peeked item into the expected return type. Strings can automatically be converted into integers and real numbers, and integers and real numbers can be converted into strings.

**Example**

```c
```
static void main(Args _args)
{
    container cnI, cnJ;
    int i, j;
    anytype aty;
    info("container cnI ...);
    cnI = ["itemBlue", "itemYellow"];  
    for (i=1; i <= conLen(cnI); i++)
        {
            aty = conPeek(cnI, i);
            info(int2str(i) + " : " + aty);
        }

    info("container cnJ ...");
    cnJ = conIns(cnI, 2, "ItemInserted");
    for (j=1; j <= conLen(cnJ); j++)
        {
            aty = conPeek(cnJ, j);
            info(int2str(j) + " : " + aty);
        }
}

/*** Output pasted from InfoLog ...
Message (10:20:03 am)
container cnI ...
1 :  itemBlue
2 :  itemYellow
container cnJ ...
1 :  itemBlue
2 :  ItemInserted
3 :  itemYellow
***/

conPoke

Modifies a container by replacing one or more of the existing elements.

Syntax

```
container conPoke(container container, int start, anytype element, ...)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>The container to modify.</td>
</tr>
<tr>
<td>start</td>
<td>The position of the first element to replace.</td>
</tr>
<tr>
<td>element</td>
<td>One or more elements to replace, separated by commas.</td>
</tr>
</tbody>
</table>

Return value

A new container that includes the new elements.

Remarks

The first element of the container is specified by the number 1.

Example
static void conPokeExample(Args _arg)
{
    container c1 = ["item1", "item2", "item3"];  
    container c2;
    int i;
    void conPrint(container c)
    {
        for (i = 1 ; i <= conLen(c) ; i++)
        {
            print conPeek(c, i);
        }
    }
    conPrint(c1);
    c2 = conPoke(c1, 2, "PokedItem");
    print "\n";
    conPrint(c2);
}
This topic describes the conversion run-time functions.

**any2Date**

Converts an *anytype* value to a *date* value.

```plaintext
date any2Date(anytype object)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>The value to convert to a date.</td>
</tr>
</tbody>
</table>

**Return value**

A *date* value.

**Remarks**

The `object` parameter can be of most data types, but useful output is obtained when it's of the *str* or *int* type. Inappropriate content generates a run-time error.

**Example**

```plaintext
static void any2DateExample(Args _args) {
    date myDate;
    str s;
    int i;
    s = "2010 6 17"; // A string object, of yyyy mm dd.
    myDate = any2Date(s);
    Global::info(strFmt("%1 is output, from input of "2010 6 17", myDate));
    i = 40361; // An int object, which represents the number of days from 1900/01/01.
    myDate = any2Date(i);
    Global::info(strFmt("%1 is output, from input of 40361", myDate));
}

/**** Infolog display.
Message (04:44:15 pm)
6/17/2010 is output, from input of "2010 6 17"
7/4/2010 is output, from input of 40361
****/```

**any2Enum**

Converts an *anytype* value to the *Name* property value of an element in the target enum.

```plaintext
enum any2Enum(anytype object)
```

**Parameters**
<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>The value to match the Value property of an element in the target enum.</td>
</tr>
</tbody>
</table>

**Return value**

The value of the Name property for whichever element in the target enum has a Value property that matches the input parameter.

**Remarks**

The object parameter can be of most data types, but useful data is obtained only when you use a parameter of the str or int type. This input object parameter refers to the Value property of an individual element in the target enum.

**Example**

```java
static void any2EnumExample(Args _args) {
    NoYes myNoYes;  // NoYes is an enum.
    int i;
    str s;
    i = 0;  // An int that will be converted.
    myNoYes = any2Enum(i);
    Global::info(strfmt("'\%1' - is the output, from input of the \%2 as int.", myNoYes, i));
    s = "1";  // A str that will be converted.
    myNoYes = any2Enum(s);
    Global::info(strfmt("'\%1' - is the output, from input of the \%2 as str.", myNoYes, s));
    /**** Infolog display.
    Message (01:05:32 pm)
    'No' - is the output, from input of the 0 as int.
    'Yes' - is the output, from input of the 1 as str.
    ****/
}
```

**any2Guid**

Converts the specified anytype object to a GUID object.

```java
guid any2Guid(anytype object)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>The value to convert to a GUID object.</td>
</tr>
</tbody>
</table>

**Return value**

A GUID object.

**any2Int**

Converts an anytype value to an int value.

```java
int any2Int(anytype object)
```
### Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>The value to convert.</td>
</tr>
</tbody>
</table>

### Return value

An int value.

### Remarks

The `object` parameter can be of most data types, but useful data is obtained only when you use parameters of the `enum`, `real`, or `str` type.

### Example

```java
static void any2IntExample(Args _args)
{
    int myInt;
    str s;
    NoYes a;
    real r;
    s = "31";
    myInt = any2Int(s);
    Global::info(strfmt("%1 is the output, from input of 31 as a str value.", myInt));
    a = NoYes::No;
    myInt = any2Int(a);
    Global::info(strfmt("%1 is the output, from input of NoYes::No as an enum value.", myInt));
    r = 5.34e2;
    myInt = any2Int(r);
    Global::info(strfmt("%1 is the output, from the input of 5.34e2 as a real value.", myInt));
}
```

/**** Infolog display.
Message (02:23:59 pm)
31 is the output, from input of 31 as a str value.
0 is the output, from input of NoYes::No as an enum value.
534 is the output, from the input of 5.34e2 as a real value.
****/

### any2Int64

Converts an anytype object to an int64 object.

```java
int64 any2Int64(anytype object)
```

### Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>The anytype object to convert.</td>
</tr>
</tbody>
</table>

### Return value

An int64 object.

### any2Real

Converts an anytype value to a real value.
real any2Real(anytype object)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>The value to convert.</td>
</tr>
</tbody>
</table>

Return value

A real value.

Remarks

The object parameter can be of most data types, but useful output is obtained for input elements of the date, int, enum, and str types.

Example

```java
static void any2RealExample(Args _args)
{
    real myReal;
    str s;
    int i;
    NoYes a;
    s = "5.12";
    myReal = any2Real(s);
    Global::info(strfmt("%1 is the output from the input of 5.12 as a str object", myReal));
    i = 64;
    myReal = any2Real(i);
    Global::info(strfmt("%1 is the output from the input of 64 as an int object", myReal));
    a = NoYes::Yes;
    myReal = any2Real(a);
    Global::info(strfmt("%1 is the output from the input of NoYes::Yes as an enum object", myReal));
}
```

/****Infolog display.
Message (02:43:57 pm)
5.12 is the output from the input of 5.12 as a str object
64.00 is the output from the input of 64 as an int object
1.00 is the output from the input of NoYes::Yes as an enum object
****/}

any2Str

Converts an anytype value to a str value.

str any2Str(anytype object)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>The value to convert.</td>
</tr>
</tbody>
</table>

Return value

A str value.

Remarks
The object parameter can be of most data types, but useful output is obtained from input elements of the date, int, and enum types.

**Example**

```java
static void any2StrExample(Args _args)
{
    str myStr;
    anytype a;
    a = "Any to string";
    myStr = any2Str(a);
    Global::info(strFmt("%1 is output, from input of Any to string as a str value", myStr));
    a = NoYes::Yes;
    myStr = any2Str(a);
    Global::info(strFmt("%1 is output, from input of NoYes::Yes as an enumeration", myStr));
}
****Infolog Display
Message (09:08:46 am)
Any to string is output, from input of Any to string as a str value
1 is output, from input of NoYes::Yes as an enumeration
****
```

anytodate

See any2Date.

anytoenum

See any2Enum.

anytoguid

See any2Guid.

anytoint

See any2Int.

anytoint64

See any2Int64.

anytoreal

See any2Real.

anytostr

See any2Str.

char2Num

Converts a character in a string to the ASCII value of the character.

```c
int char2Num(str text, int position)
```
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>The string that contains the character.</td>
</tr>
<tr>
<td>position</td>
<td>The position of the character in the string.</td>
</tr>
</tbody>
</table>

**Return value**

The ASCII value of the character as an `int` object.

**Remarks**

```plaintext
char2Num("ABCD\EFG",3); //Returns the numeric value of C, which is 67.
char2Num("ABCD\EFG",1); //Returns the numeric value of A, which is 65.
```

### date2Num

Converts a date to an integer that corresponds to the number of days since January 1, 1900.

```plaintext
int date2Num(date _date)
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_date</td>
<td>The date to convert.</td>
</tr>
</tbody>
</table>

**Return value**

The number of days between January 1, 1900, and the specified date.

**Example**

```plaintext
//Returns the value377.
date2Num(1311901);
static void date2NumExample(Args _arg)
{
    date d = today();
    int i;
    i = date2Num(d);
    print i;
}
```

### date2Str

Converts the specified date to a string.

```plaintext
str date2Str(date date, int sequence, int day, int separator1, int month, int separator2, int year [, int flags = DateFlags::None])
```

**Parameters**
<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>The date to convert.</td>
</tr>
<tr>
<td>sequence</td>
<td>A three-digit number that indicates the sequence for the components of the date: 1 for day, 2 for month, and 3 for year.</td>
</tr>
<tr>
<td>day</td>
<td>An enumeration value that indicates the format for the day component of the date.</td>
</tr>
<tr>
<td>separator1</td>
<td>An enumeration value that indicates the separator to use between the first two components of the date.</td>
</tr>
<tr>
<td>month</td>
<td>An enumeration value that indicates the format for the month component of the date.</td>
</tr>
<tr>
<td>separator2</td>
<td>An enumeration value that indicates the separator to use between the last two components of the date.</td>
</tr>
<tr>
<td>year</td>
<td>An enumeration value that indicates the format for the year component of the date.</td>
</tr>
<tr>
<td>flags</td>
<td>A <code>DateFlags</code> enumeration value that indicates whether the language settings on the local computer should be used to calculate the proper left-to-right or right-to-left sequence in the returned string.</td>
</tr>
</tbody>
</table>

**Return value**
A string that represents the specified date.

**Remarks**
MorphX allocates valid values to the formatting parameters if the specified values aren't valid. To use the date format that the user specified in Regional Settings, use the `strFmt` or `date2Str` function and specify -1 in all the formatting parameters. When the regional settings control the date format, the settings can change from user to user. If -1 is used for either `separator` parameter, both separators default to Regional Settings. The `sequence` parameter values must be any three-digit number that contains exactly one occurrence of each the digits 1, 2, and 3. The digits 1, 2, and 3 represent day, month, and year, respectively. For example, 321 produces the sequence year, month, and day. Or the value can be -1 to use Regional Settings. No enumeration type should be used for this parameter, because numbers such as 321 exceed the range of valid values for enumeration values, which is 0 through 250, inclusive. The default value of the `flags` parameter is the `DateFlags::None` enumeration value, which means no left-to-right or right-to-left sequence processing is done.

**Example**
The following example displays the current date in the sequence of year, month, and day.
static void Job2(Args _args)
{
    date currentDate = today();
    str s;
    int iEnum;
    s = date2Str
    (currentDate,
     321,
     DateDay::Digits2,
     DateSeparator::Hyphen, // separator1
     DateMonth::Digits2,
     DateSeparator::Hyphen, // separator2
     DateYear::Digits4
    );
    info("Today is:  " + s);
}
/** Example Infolog output
Message (12:36:21 pm)
Today is:  2009-01-13
**/

datetime2Str
Converts a \texttt{utcdatetime} value into a string.

\begin{verbatim}
str datetime2Str(utcdatetime datetime [, int flags = DateFlags::None])
\end{verbatim}

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>datetime</td>
<td>The \texttt{utcdatetime} value to convert.</td>
</tr>
<tr>
<td>flags</td>
<td>A \texttt{DateFlags} enumeration value that indicates whether to use local settings for right-to-left output.</td>
</tr>
</tbody>
</table>

\textbf{Return value}
A string that represents the \texttt{utcdatetime} value that was specified as the \texttt{datetime} parameter.

\textbf{Remarks}

\textbf{Null date-time input}
If the minimum \texttt{utcdatetime} value is specified for the \texttt{datetime} parameter, the \texttt{datetime2Str} function treats it as a null input value. This causes the function to return an empty string. The date-time \texttt{1900-01-01T00:00:00} is returned by the \texttt{DateTimeUtil::minValue} method. This minimum value is treated as null.

\textbf{Right-to-left local settings}
The default behavior of this function is to generate the string in left-to-right sequence, where the year portion is leftmost. However, the \texttt{flags} parameter value of the \texttt{DateFlags::FormatAll} enumeration value directs the function to generate the string in right-to-left sequence if the local settings are configured for right-to-left. The format of the \texttt{toStr} method of the \texttt{DateTimeUtil} class is unaffected by regional settings.

\textbf{Example}
static void jobTestDatetime2str( Args _args )
{
    utcdatetime utc2 = 1959-06-17T15:44:33;
    str s3;
    s3 = datetime2Str( utc2 );
    info( s3 );
}

enum2Str
Converting the specified enumerated text to a character representation.

str enum2Str(enum enum)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum</td>
<td>The enumerated text to convert.</td>
</tr>
</tbody>
</table>

Return value
The value of the enumeration as a string.

Example
The following example returns the string "Not included." This is the label for the IncludeNot value of the ListCode enumeration type.

static void enum2StrExample(Args _arg)
{
    ListCode l;
    l = ListCode::IncludeNot;
    print enum2Str(l);
}

guid2Str
Converting the specified GUID object to the equivalent string.

str guid2String(guid _uuid)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_uuid</td>
<td>The GUID object to convert.</td>
</tr>
</tbody>
</table>

Return value
The string equivalent of the specified GUID object.

Example
static void guid2StrExample()
{
  guid _guid;
  str stringGuid;
  _guid = Global::guidFromString("{12345678-1234-1234-1234-123456789abc}");
  print strfmt("GUID is %1", _guid);
  stringGuid = guid2str(_guid);
  info("String GUID is "+stringGuid);
}
/**** Output to Infolog
String GUID is {12345678-1234-1234-1234-123456789ABC}
****/

### int2Str

Converts an integer to the equivalent string.

str int2Str(int integer)

#### Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>The integer to convert.</td>
</tr>
</tbody>
</table>

#### Return value

A string representation of the integer.

#### Example

```cpp
static void int2StrExample(Args _arg)
{
  print "This is int2Str, value is "+int2Str(intMax());
  print "This is int642Str, value is "+int642Str(int64Max());
}
```

### int642Str

Converts the specified `integer` parameter to the equivalent text string.

str int642Str(int64 integer)

#### Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>The int64 to convert to a string.</td>
</tr>
</tbody>
</table>

#### Return value

The equivalent text string of the `integer` parameter.

#### Example

```cpp
static void int642StrExample(Args _arg)
{
  print "This is int642Str, value is "+int642Str(int64Max());
}
```
static void example()
{
    print "This is int2Str, value is " + int2Str(intMax());
    print "This is int642Str, value is " + int642Str(int64Max());
}

num2Char

Converts an integer to the corresponding ASCII character.

str num2Char(int figure)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>figure</td>
<td>The integer to convert to a character.</td>
</tr>
</tbody>
</table>

Return value

The character that is represented by the specified integer.

Example

static void num2CharExample(Args _arg)
{
    str s;
    s = num2Char(42);
    // Prints an asterisk * -the character represented by 42.
    print s;
}

num2Date

Retrieves the date that corresponds to the specified number of days after January 1, 1900.

date num2Date(int _days)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_days</td>
<td>The number of days after January 1, 1900 to return the date for. Note: The first valid date is January 1, 1901. Therefore, the num2Date function doesn't return a valid date unless _days is more than 365.</td>
</tr>
</tbody>
</table>

Return value

The date that is the number of days that is specified by the _days parameter after January 1, 1900.

Remarks

num2Date(366); //Returns the date 01/01/1901 (1 January 1901).
num2Str

Converts a real number to a string.

\[
\text{str num2Str(\text{real number, int character, int decimals, int separator1, int separator2})}
\]

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>The real number to convert to a string.</td>
</tr>
<tr>
<td>character</td>
<td>The minimum number of characters that are required in the text.</td>
</tr>
<tr>
<td>decimals</td>
<td>The required number of decimal places.</td>
</tr>
<tr>
<td>separator1</td>
<td>A <code>DecimalSeparator</code> enumeration value.</td>
</tr>
<tr>
<td>separator2</td>
<td>A <code>ThousandSeparator</code> enumeration value.</td>
</tr>
</tbody>
</table>

**Return value**

A string that represents the number.

**Remarks**

For the `decimals` parameter, the maximum value is 16. If a larger number is used, this method obtains a value for the `decimals` parameter from the local computer instead. In both cases, rounding occurs. Here are the possible enumeration values for the `separator1` parameter:

- 99 – Auto (the formatting settings of the user determine what decimal separator is used), enumeration value `DecimalSeparator::Auto`
- 1 – Dot (.), enumeration value `DecimalSeparator::Dot`
- 2 – Comma (,), enumeration value `DecimalSeparator::Comma`

Here are the possible values for the `separator2` parameter:

- 99 – Auto (the formatting settings of the user determine what thousand separator is used)
- 0 – None (no thousand separator), enumeration value `ThousandSeparator::None`
- 1 – Dot (.), enumeration value `ThousandSeparator::Dot`
- 2 – Comma (,), enumeration value `ThousandSeparator::Comma`
- 3 – Apostrophe (‘), enumeration value `ThousandSeparator::Apostrophe`
- 4 – Space ( ), enumeration value `ThousandSeparator::Space`

**Example**

In the following code example, the first call to the `num2str` method provides 16 for the `decimals` parameter, and the second call provides 17.

```java
static void Job_Num2Str(Args _args)
{
    real realNum = 0.1294567890123456777; // 19 decimals places.
    info(num2Str(realNum, 0, 16, DecimalSeparator::Dot, ThousandSeparator::Space)); // 16 decimal places
    info(num2Str(realNum, 0, 17, DecimalSeparator::Dot, ThousandSeparator::Space)); // 17 decimal places
}
```
str2Date

Converts the specified string to a date value.

date str2Date(str _text, str _sequence)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_text</td>
<td>The string to convert to a date value.</td>
</tr>
<tr>
<td>_sequence</td>
<td>A three-digit integer that describes the positions of the day, month, and year in the string to convert.</td>
</tr>
</tbody>
</table>

Return value

A date value.

Remarks

Use the following values to specify the positions of the day, month, and year in the _sequence parameter:

- **Day**: 1
- **Month**: 2
- **Year**: 3

For example, if the sequence in the string is month, year, and then day, the _sequence parameter must be **231**. A 0 (zero) date is returned if the input parameters specify an invalid date. The following two examples specify an invalid date.

str2Date("31/12/44", 123) // Year must be four digits to reach the minimum of January 1 1901.  
str2Date("31/12/2044", 213) // 213 means the month occurs first in the string, but 31 cannot be a month.

Example

```java
static void str2DateExample(Args _arg)
{
    date d;
    d = str2Date("22/11/2007", 123);
    print d;
}
```

str2Datetimetime

Generates a utcdatetime value from the specified string of date and time information.
utcdatetime str2datetime( str text, int sequence )

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>The string to convert to a utcdatetime value.</td>
</tr>
<tr>
<td>sequence</td>
<td>A three-digit number that describes the sequence of the date components in the text parameter.</td>
</tr>
</tbody>
</table>

Return value

A utcdatetime value that represents the specified date and time.

Remarks

The syntax requirements for the date portion of the text parameter are flexible. The variety of valid formats is the same as in the date2str function. Each of the following calls to str2datetime is valid, and all of them produce the same output.

\[
\begin{align*}
\text{utc3} &= \text{str2datetime( "1985/02/25 23:04:59" ,321 );} \\
\text{utc3} &= \text{str2datetime( "Feb-1985-25 11:04:59 pm" ,231 );} \\
\text{utc3} &= \text{str2datetime( "2 25 1985 11:04:59 pm",123 );}
\end{align*}
\]

Each component of the date time is represented by a digit in the sequence parameter:

- 1 – Day
- 2 – Month
- 3 – Year

For example, year, month, day order is 321. All valid values contain each of these three digits exactly one time. If the value of the sequence parameter isn’t valid, the regional settings are used to interpret the input text parameter. If the input parameters describe an invalid date and time, an empty string is returned.

Example

```c
static void JobTestStr2datetime( Args _args )
{
    utcdatetime utc3;
    str sTemp;
    utc3 = str2datetime( "1985/02/25 23:04:59",321 );
    sTemp = datetime2str( utc3 );
    print( "sTemp == " + sTemp );
}
```

str2Enum

Retrieves the enum element for which the localized Label property value matches the input string.

```
enum str2Enum(enum _type, str _text)
```
### Return value

An element of the target enum, which also represents an int.

### Remarks

The related function `enum2str` returns the value of a Label property from one element in the enum. The value that is returned by `enum2str` function can be the input for the `_type` parameter of the `str2enum` function. An appropriate value for the `_text` parameter is `enum2Str(BankAccountType::SavingsAccount)`. Each element of an enum has a Name property and a Label property. In a fresh install, the Name values are almost always English words. In the English edition, the Label property value is almost always the same as the Name value. However, in non-English editions, the Label values are localized and therefore don’t match the Name values.

### Example

To avoid string mismatches that are caused by localization to other spoken languages, we recommend that you use the `enum2str` function to generate the input into the `str2enum` function. The following example shows the appropriate way to use the `str2enum` function together with the `enum2str` function.

```csharp
static void str2Enum_AcrossLangs(Args _arg)
{
    BankAccountType bat;
    str sEnumValueLabelLocalized;
    int nInt;
    // enum2str.
    sEnumValueLabelLocalized = enum2str(BankAccountType::SavingsAccount);
    info("Localized friendly string: ");
    + sEnumValueLabelLocalized);
    // str2enum.
    bat = str2Enum(bat, sEnumValueLabelLocalized);
    nInt = bat;
    info("nInt = " + int2str(nInt));
    /********** Actual output:**********/
    Message (04:32:12 pm)
    Localized friendly string: Savings account
    nInt = 1
    /**********/}
```

---

### str2Guid

Converts a string to a GUID object.

#### Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>guid</td>
<td>A string that represents a GUID.</td>
</tr>
</tbody>
</table>

#### Return value
A GUID that is represented by the input string.

**Remarks**
For example, a valid value for the `guid` parameter is `{12345678-1234-abCD-3456-123456789012}`, either with or without the braces.

**str2Int**
Converts a string to the equivalent integer.

```
int str2Int(str _text)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_text</td>
<td>The string to convert to an integer.</td>
</tr>
</tbody>
</table>

**Return value**
The integer equivalent of the specified string.

**Example**

```
static void str2IntExample(Args _arg)
{
  int i;
  i = str2Int("1234567890");
  print "i = " + int2Str(i);
}
```

**str2Int64**
Converts a string into an `Int64` value.

```
int str2Int64(str text)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>The string to convert.</td>
</tr>
</tbody>
</table>

**Return value**
The `Int64` value of the specified string.

**Example**
static void str2Int64Example(Args _args)
{
    str myStr;
    str tooBig;
    Int64 myInt64;
    myStr = "1234567890";
    tooBig = int642str(int64Max()+1);
    myInt64 = str2Int64(myStr);
    print strfmt ("int64: %1",myInt64);
    myInt64 = str2Int64(tooBig);
    print strfmt ("Too big int64: %1",myInt64);
}

str2Num

Converts a string to a real number.

real str2Num(str _text)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_text</td>
<td>The string to convert to a real number</td>
</tr>
</tbody>
</table>

Return value

The real number if the specified string contains a valid number; otherwise, 0 (zero).

Remarks

The following examples show how this function is used.

- str2Num("123.45") returns the value 123.45.
- str2Num("a123") returns the value 0.0.
- str2Num("123a") returns the value 123.00.

Scanning occurs from left to right and ends when a character can't be converted to part of a real number.

Example
static void str2NumToReal(Args _arg)
{
    real r;
    str s;
    r = str2Num("3.15");
    s = strFmt("r = %1", r);
    info(s);
}
/*** Infolog output. 
Message @SYS14327 (02:36:12 pm)
r = 3.15
***/

static void str2NumExponentialSyntax(Args _args)
{
    Qty qty1, qty2, qty3;
    qty1 = str2num('1e-3'); // Bad syntax by the user.
    qty2 = str2num('1.e-3');
    qty3 = str2num('1.0e-3');
    info(strfmt('Result: %1; Expected: %2', num2str(qty1, 0,3,2,0), '0.001'));
    info(strfmt('Result: %1; Expected: %2', num2str(qty2, 0,3,2,0), '0.001'));
    info(strfmt('Result: %1; Expected: %2', num2str(qty3, 0,3,2,0), '0.001'));
}
/*** Infolog output. The first result differs from expectations. 
Message @SYS14327 (02:20:55 pm) 
Result: 1,000; Expected: 0.001 
Result: 0,001; Expected: 0.001
Result: 0,001; Expected: 0.001
***/

str2Time

Converts a string to a timeOfDay value.

int str2Time(str _text)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_text</td>
<td>The time to use to calculate the number of seconds since midnight.</td>
</tr>
</tbody>
</table>

Return value

The number of seconds between midnight and the _text parameter; otherwise, -1.

Remarks

str2Time("05:01:37") //Returns the value 18097.
str2Time("7 o’clock") //Returns the value -1.

Example
```csharp
static void str2TimeExample(Args _arg)
{
    int i;
    i = str2Time("11:30");
    print i;
}
```

time2Str

Converts a `timeOfDay` value to a string that includes hours, minutes, and seconds.

```csharp
str time2Str(int _time, int _separator, int _timeFormat)
```

### Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_time</td>
<td>A <code>timeOfDay</code> value.</td>
</tr>
<tr>
<td>_separator</td>
<td>A <code>TimeSeparator</code> enumeration value that indicates the characters between the hours, minutes, and seconds in the output string.</td>
</tr>
<tr>
<td>_timeFormat</td>
<td>A <code>TimeFormat</code> enumeration value that indicates whether a 12-hour clock or a 24-hour clock is used.</td>
</tr>
</tbody>
</table>

### Return value

A string that represents the specified time.

### Remarks

The value of the `_time` parameter is the number of seconds since midnight.

### Example

```csharp
static void TimeJob4(Args _args)
{
    timeOfDay theTime = timeNow();
    info( time2Str( theTime, TimeSeparator::Colon, TimeFormat::AMPM ) );
}
/**
Message (04:33:56 pm)
04:33:56 pm
**/
```

uint2Str

Converts an integer to a string. The assumption is that the integer is unsigned.

```csharp
str uint2Str(int integer)
```

### Parameters
<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>The integer to convert.</td>
</tr>
</tbody>
</table>

**Return value**
The string equivalent to the specified unsigned integer.

**Remarks**
Use this function instead of the `int2str` function for very large integers, such as record IDs.

```plaintext
info(int2str(3123456789)); //returns -1171510507 as a string.
info(uint2str(3123456789)); //returns 3123456789 as a string.
```
This topic describes the date run-time functions.

**dayName**

Retrieves the name of the day of the week that is specified by a number.

```cpp
str dayName(int number)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>The number of a day in a week.</td>
</tr>
</tbody>
</table>

**Return value**

The day of the week specified by the number parameter.

**Remarks**

The valid values for the number parameter are 1 through 7. Monday is represented by 1, Tuesday by 2, and Sunday by 7.

**Example**

```cpp
static void dayNameExample(Args _arg)
{
    str s;
    s = dayName(01);
    print "First day of the week's name is " + s;
    pause;
}
```

**dayOfMth**

Calculates the number of the day in the month for the specified date.

```cpp
int dayOfMth(date date)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>The date to test.</td>
</tr>
</tbody>
</table>

**Return value**

An integer between 1 and 31 that indicates the day of the month for the specified date.
Remarks

dayOfMth(31122001) //returns 31.

Example

```java
static void dayOfMthExample(Args _arg)
{
    date d = today();
    int i;
    i = dayOfMth(d);
    print "Today's day of the month is " + int2Str(i);
    pause;
}
```

dayOfWk

Calculates the number of day in the week for the specified date. **Note:** Monday is represented by 1, Tuesday by 2, and Sunday by 7.

```java
int dayOfWk(date date)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>A date value that indicates the year, month, and day.</td>
</tr>
</tbody>
</table>

Return value

The number of the specified day in the week.

Example

```java
static void dayOfWkExample(Args _arg)
{
    date d = today();
    int i;
    i = dayOfWk(d);
    print "Today's day of the week is " + int2Str(i);
    pause;
}
```

dayOfYr

Calculates the number of days between January 1 and the specified date.

```java
int dayOfYr(date _date)
```

Parameters
### dayOfYr

A date that specifies the year, month, and day.

**Return value**
The number of days between January 1 and the specified date, inclusive.

**Remarks**
January 1 is 1, and December 31 is either 365 or 366.

**Example**

```java
static void dayOfYrExample(Args _arg)
{
    date d = today();
    int i;
    i = dayOfYr(d);
    print "Today's day of the year is " + int2Str(i);
    pause;
}
```

### endMth

Calculates the last date in the month of the specified date.

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>A date value that indicates a year, month, and day.</td>
</tr>
</tbody>
</table>

**Return value**
The date value of the last day in the specified month.

**Remarks**
endMth(0221988); //Returns the date 2921988 because 1988 is a leap year.
endMth(0221989); //Returns the date 2821989.

### mkDate

Creates a date, based on three integers that indicate the day, month, and year, respectively. "Shorthand" values for the year argument, for example, "y", are not supported.

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>int day</td>
<td></td>
</tr>
<tr>
<td>int month</td>
<td></td>
</tr>
<tr>
<td>int year</td>
<td></td>
</tr>
</tbody>
</table>
### mkDate

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>day</td>
<td>An integer that represents the day of the month.</td>
</tr>
<tr>
<td>month</td>
<td>An integer that represents the month of the year.</td>
</tr>
<tr>
<td>year</td>
<td>An integer that represents the year, which must be between 1900 and 2154.</td>
</tr>
</tbody>
</table>

**Return value**

A *date* value that is based on the values of the *day*, *month*, and *year* parameters.

**Remarks**

If the date isn't valid, this method returns a 0 (zero, 1/1/1900) date. Beginning with Dynamics AX 7.0 (February 2016), shortcut values for the year, e.g. 75 for 1975, are not supported. If you provide a shortcut value for the year, a date of 1/1/1900 is returned.

**Example**

```csharp
static void mkDateExample(Args _arg)
{
    date d;
    // Returns the date 01/01/2005.
    d = mkDate(1, 1, 2005);
    print d;
    pause;
}
```

### mthName

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>The number of the month.</td>
</tr>
</tbody>
</table>

**Return value**

The name of the specified month.

**Remarks**

The valid values of the *number* parameter are 1 through 12. January is represented by 1 and December by 12.

**Example**
static void mthNameExample(Args _arg)
{
    str s;
    // MthName(6) returns the text string "June".
    s = mthName(6);
    print "Month name is " + s;
    pause;
}

mthOfYr

Retrieves the number of the month in the year for the specified date. **Note:** January is 1, February is 2, and December is 12.

```
int mthOfYr(date date)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>A date that specifies a year, month, and day.</td>
</tr>
</tbody>
</table>

**Return value**

The number of the month in the year, for the month that is represented by the *date* parameter.

**Example**

```
static void mthOfYrExample(Args _arg)
{
    int i;
    i = mthOfYr(today());
    print "The number of the month in today's date is " + int2Str(i);
    pause;
}
```

nextMth

Retrieves the date in the following month that corresponds most closely to the specified date.

```
date nextMth(date date)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>The date to match in the following month.</td>
</tr>
</tbody>
</table>

**Return value**

The closest match to the specified date that is found in the next month.

**Remarks**
nextMth(2921996); //returns 29/03/1996.
nextMth(3111996); //returns 2921996, because 1996 is a leap year.

Example

```java
static void nextMthExample(Args _arg)
{
    date d;
    d = nextMth(today());
    print "Closest date next month is "
        + date2Str(d, 2, 2, -1, 2, -1, 4);
    pause;
}
```

nextQtr

Retrieves the date in the following quarter that corresponds most closely to the specified date.

```java
date nextQtr(date date)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>The date to match in the following quarter.</td>
</tr>
</tbody>
</table>

Return value

The closest match to specified date that is found in the next quarter.

Remarks

For example, `nextQtr(3111998)` returns `3041998`.

Example

```java
static void nextQtrExample(Args _arg)
{
    date d;
    d = nextQtr(today());
    print "Closest date next quarter is "
        + date2Str(d, 2, 2, -1, 2, -1, 4);
    pause;
}
```

nextYr

Retrieves the date in the following year that corresponds most closely to the specified date.

```java
date nextYr(date date)
```

Parameters
<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>The date to match in the following year.</td>
</tr>
</tbody>
</table>

**Return value**
The closest match to the specified date that is found in the following year.

**Remarks**
For example, `nextYr(2921998)` returns 2821999.

**Example**
```java
static void nextYrExample(Args _arg)
{
    date d;
    d = nextYr(today());
    print "Closest date next year is "
        + date2Str(d, 2, 2, -1, 2, -1, 4);
    pause;
}
```

**prevMth**
Retrieves the date in the previous month that corresponds most closely to the specified date.

```java
date prevMth(date date)
```

**Parameters**
<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>The date to match in the previous month.</td>
</tr>
</tbody>
</table>

**Return value**
The closest match to the specified date that is found in the previous month.

**Remarks**
 prevMth(3131996); //Returns the date 29/02/1996 because 1996 is a leap year.
 prevMth(2821998); //Returns the date 28/01/1998.

**prevQtr**
Retrieves the date in the previous quarter that corresponds most closely to the specified date.

```java
date prevQtr(date date)
```

**Parameters**
<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>The date to match in the previous quarter.</td>
</tr>
</tbody>
</table>
Return value
The closest match to the specified date that is found in the previous quarter.

Remarks
prevQtr(3041998); //Returns the date 30/01/1998.
prevQtr(2951996); //Returns the date 29/02/1996, because 1996 is a leap year.

prevYr
Retrieves the date in the previous year that corresponds most closely to the specified date.

date prevYr(date date)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>The date to match in the previous year.</td>
</tr>
</tbody>
</table>

Return value
The closest match to the specified date that is found in the previous year.

Remarks
prevYr(2921996); //Returns the date 28/02/1995 because 1996 is a leap year.
prevYr(2821998); //Returns the date 28/02/1997.

systemDateGet
Retrieves the session date, if it has been set.

date systemDateGet()

Return value
The session date if it has been set; otherwise, the system date.

Remarks
Consider using Session date and time on the Tools menu to open the Session date and time page. This page can be used to actively set the session date. After this set action is detected by the system, subsequent calls to the systemDateGet function return the session date. The today function returns the system date. This function doesn't support time zones.

Example
The following example shows the date in the Infolog window.
static void Job_systemDateGet(Args _arg)
{
    info( date2Str(
        systemDateGet(),       // X++ language function.
        321,                   // 321 = ymd
        DateDay::Digits2,
        DateSeparator::Hyphen, // separator1
        DateMonth::Digits2,
        DateSeparator::Hyphen, // separator2
        DateYear::Digits4
    ));
}

/*********** Actual Infolog output
Message (03:46:00 pm)
2012-04-16
***********/

systemDateSet
Changes the system date.

date systemDateSet(date _date)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_date</td>
<td>The new date for the system.</td>
</tr>
</tbody>
</table>

Return value
The new system date.

Remarks
This function doesn’t affect the session date. This method changes the date, but the time will be set to 0 (zero).

Example
The following example sets the system date to today’s date.

static void systemDateSetExample(Args _arg)
{
    date d = today();
    d = systemDateSet(d);
    print d;
}

timeNow
Retrieves the current system time.

int timeNow()
Example

```java
static void timeNowExample(Args _arg)
{
    int i;
    i = timeNow();
    print "The number of seconds since midnight is " + int2Str(i);
    pause;
}
```

today

Retrieves the current date on the system.

```java
date today()
```

Return value

The current date.

Example

```java
static void todayExample(Args _arg)
{
    date d;
    d = today();
    print "Today's date is " + date2Str(d, 0, 2, -1, 2, -1, 4);
    pause;
}
```

wkOfYr

Calculates the week of the year that a date falls in, according to the ISO 8601 specification.

```java
int wkOfYr(date _date)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_date</td>
<td>The date to calculate the week of the year for</td>
</tr>
</tbody>
</table>

Return value

The sequence number of the week that the `_date` parameter occurs in.

Example

The following code example compares the `wkOfYr` function with the `Global::weekOfYear` method. The function and the method produce different results.
// X++ job, under AOT > Jobs.
static void WeekTests3Job(Args _args)
{
    int weekNum, i;
    date dateTest;
    str sMessages[];
    //---------------------------------------------
    sMessages[1] = "----- #1. For Sunday, January 5, 2003 -----";
    dateTest = 512003; // DayMonthYear format.
    weekNum = wkOfYr(dateTest);
    sMessages[2] = int2str(weekNum) + " = wkOfYr function";
    weekNum = Global::weekOfYear(dateTest);
    sMessages[3] = int2str(weekNum) + " = Global::weekOfYear method";
    //---------------------------------------------
    sMessages[4] = " ";
    sMessages[5] = "----- #2. For Wednesday, August 20, 2003 -----";
    dateTest = 2082003;
    weekNum = wkOfYr(dateTest);
    sMessages[6] = int2str(weekNum) + " = wkOfYr function";
    weekNum = Global::weekOfYear(dateTest);
    sMessages[7] = int2str(weekNum) + " = Global::weekOfYear method";
    //---------------------------------------------
    sMessages[8] = " ";
    dateTest = 28122003;
    weekNum = wkOfYr(dateTest);
    sMessages[10] = int2str(weekNum) + " = wkOfYr function";
    weekNum = Global::weekOfYear(dateTest);
    sMessages[11] = int2str(weekNum) + " = Global::weekOfYear method";
    for (i=1; i<= 11; i++)
    {
        Global::info(sMessages[i]);
    }
}

The previous example sent the following information to the Infolog for display. The output shows that there are differences between wkOfYr and Global::weekOfYear.

Message (01:59:13 pm) ----- 
#1. For Sunday, January 5, 2003 ----- 1 = wkOfYr function 2 = Global::weekOfYear method ----- 
#2. For Wednesday, August 20, 2003 ----- 34 = wkOfYr function 34 = Global::weekOfYear method ----- 
#3. For Sunday, December 28, 2003 ----- 52 = wkOfYr function 1 = Global::weekOfYear method

year
Retrieves the year from a date value.

int year(date _date)

Parameters

PARAMETER DESCRIPTION
_date The date to return the year from.

Return value
The year of the specified date.

Remarks
year(0221998); //Returns the value 1998.
This topic describes the math run-time functions.

These functions perform mathematical calculations.

**abs**

Retrieves the absolute value of a real number. Examples:

- abs(-100.0) returns the value 100.0.
- abs(30.56) returns the value 30.56.

**Syntax**

real abs(real arg)

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg</td>
<td>The number to get the absolute value of.</td>
</tr>
</tbody>
</table>

**Return value**

The absolute value of *arg*.

**Example**

```cpp
static void absExample(Args _args)
{
    real r1;
    real r2;
    ;
    r1 = abs(-3.14);
    r2 = abs(3.14);
    if (r1 == r2)
    {
        print "abs of values are the same";
        pause;
    }
}
```

**acos**

Retrieves the arc cosine of a real number.

**NOTE**

Argument values that are outside the -1 to 1 range cause the following run-time error: "Argument for trigonometric function out of range."
Syntax

real acos(real arg)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg</td>
<td>The number to retrieve the arc cosine of</td>
</tr>
</tbody>
</table>

Return value

The arc cosine of arg.

Example

```java
static void acosExample(Args _args)
{
    real r;
    str s;
    r = acos(0.0);
    s = strFmt("The arc cosine of 0.0 is %1 ", r);
    print s;
    pause;
}
```

asin

Retrieves the arc sine of a real number.

NOTE

Argument values that are outside the -1 to 1 range cause the following run-time error: “Argument for trigonometric function out of range.”

Syntax

real asin(real arg)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg</td>
<td>The number to calculate the arc sine for</td>
</tr>
</tbody>
</table>

Return value

The arc sine of the specified number.

Remarks

aSin(0.36) returns 0.37.

atan

Retrieves the arc tangent of a real number.
Syntax

```
real atan(real arg)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg</td>
<td>The number to calculate the arc tangent for.</td>
</tr>
</tbody>
</table>

Return value

The arc tangent of the specified number.

Remarks

`atan(0.36)` returns 0.35.

Example

```
static void atanExample(Args _args)
{
    real r;
    r = atan(1.0);
    print strFmt("The Arc Tangent of 1.0 is %1", r);
    pause;
}
```

corrFlagGet

Retrieves the state of the correction flag for a real number.

Syntax

```
int corrFlagGet(real arg)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg</td>
<td>The flag to retrieve the state for.</td>
</tr>
</tbody>
</table>

Return value

A non-zero value if the flag is set; 0 (zero) if the flag is cleared.

Example

```
static void corrFlagGetExample(Args _args)
{
    real rr;
    rr = corrFlagSet(0.36,2);
    print(corrFlagGet(rr));
}
```
corrFlagSet
Controls the correction flag for a real number.

Syntax

```
real corrFlagSet(real real, int arg)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>real</td>
<td>The number in which to turn the correction flag on or off.</td>
</tr>
<tr>
<td>arg</td>
<td>0 to turn the flag off; a non-zero value to turn the flag on.</td>
</tr>
</tbody>
</table>

Return value
0 if the flag is now off; a non-zero value if the flag is now on.

COS
Retrieves the cosine of a real number.

Syntax

```
real cos(real arg)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg</td>
<td>The number to find the cosine for.</td>
</tr>
</tbody>
</table>

Return value
The cosine of the specified number.

Remarks
The value of the arg parameter must be in radians.

Example
The following code example displays 0.76.

```
static void cosExample(Args _arg)
{
    real r;
    r = cos(15);
    print strFmt("Cos of 15 is %1", r);
    pause;
}
```

cosh
Retrieves the hyperbolic cosine of a real number.
Syntax

real cosh(real arg)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg</td>
<td>The hyperbolic number to calculate the cosine for.</td>
</tr>
</tbody>
</table>

Return value

The hyperbolic cosine of the specified number.

Remarks

The value of the arg parameter must be in radians.

Example

```java
static void coshExample(Args _arg)
{
    real r;
    r = cosh(0.1);
    print "The hyperbolic cosine of 0.1 is " + num2Str(r, 2, 2, 1, 1);
    pause;
}
```

decRound

Rounds a number to the specified number of decimal places.

Syntax

real decRound(real figure, int decimals)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>figure</td>
<td>The number to round.</td>
</tr>
<tr>
<td>decimals</td>
<td>The number of decimal places to round to.</td>
</tr>
</tbody>
</table>

Return value

The value of the specified number, rounded to the specified number of decimal places.

Remarks

The value of the decimals parameter can be positive, 0 (zero), or negative.
exp

Retrieves the natural antilogarithm of the specified real number.

**Syntax**

```plaintext
real exp(real arg)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg</td>
<td>The real number to calculate the natural antilogarithm for.</td>
</tr>
</tbody>
</table>

**Return value**

The natural antilogarithm of the specified real number.

**Remarks**

The calculated natural antilogarithm is the natural logarithm $e$ raised to the power that is indicated by the `arg` parameter.

**Example**

```java
static void expExample(Args _arg)
{
    real r1;
    real r2;
    r1 = exp(2.302585093);
    r2 = exp10(2.302585093);
    print strFmt("exp of 2.302585093 is %1", r1);
    print strFmt("exp10 of 230258 is %1", r2);
    pause;
}
```

exp10

Retrieves the base-10 antilogarithm of the specified real number.

**Syntax**

```plaintext
real exp10(real decimal)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>decimal</td>
<td>The real number to calculate the base-10 antilogarithm for.</td>
</tr>
</tbody>
</table>

- `decRound(1234.6574,2)` returns the value `1234.66`.
- `decRound(1234.6574,0)` returns the value `1235`.
- `decRound(1234.6574,-2)` returns the value `1200`.
- `decRound(12345.6789,1)` returns the value `12345.70`.
- `decRound(12345.6789,-1)` returns the value `12350.00`.
Return value
The 10-based antilogarithm of the value of the decimal parameter.

Example

```c
static void exp10Example(Args _arg)
{
    real r1;
    real r2;
    r1 = exp(2.302585093);
    r2 = exp10(2.302585093);
    print strFmt("exp of 2.302585093 is %1", r1);
    print strFmt("exp10 of 230258 is %1", r2);
    pause;
}
```

frac
Retrieves the decimal part of a real number.

Syntax

```c
real frac(real decimal)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>decimal</td>
<td>The real number to retrieve the decimal part for</td>
</tr>
</tbody>
</table>

Return value
The decimal part of the specified number.

Remarks
frac(12.345) returns the value 0.345.

log10
Retrieves the 10-digit logarithm of a real number.

Syntax

```c
real log10(real arg)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg</td>
<td>The number to calculate the logarithm for</td>
</tr>
</tbody>
</table>

Return value
The base-10 logarithm of the specified number.

Remarks
**log10(200)** returns the value **2.30**.

**logN**
Retrieves the natural logarithm of the specified real number.

**Syntax**
```plaintext
real logN(real arg)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg</td>
<td>The number to calculate the natural logarithm for.</td>
</tr>
</tbody>
</table>

**Return value**
The natural logarithm of the specified number.

**Remarks**

**logN(45)** returns the value **3.81**.

**max**
Retrieves the larger of two specified values.

```plaintext
anytype max(anytype object1, anytype object2)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>object1</td>
<td>The first value.</td>
</tr>
<tr>
<td>object2</td>
<td>The second value.</td>
</tr>
</tbody>
</table>

**Return value**
The larger of the two values that are specified by the `object1` and `object2` parameters.

**Remarks**

- **max(12.0,12.1)** returns the value **12.1**.
- **max(2,33)** returns the value **33**.

**min**
Retrieves the smaller of two specified values.

```plaintext
anytype min(anytype object1, anytype object2)
```

**Parameters**
### min

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>object1</td>
<td>The first value.</td>
</tr>
<tr>
<td>object2</td>
<td>The second value.</td>
</tr>
</tbody>
</table>

**Return value**

The smaller of the two values that are specified by the `object1` and `object2` parameters.

**Remarks**

\( \text{min}(2, 33) \) returns the value 2.

**Example**

```java
static void minExample(Args _arg)
{
    anytype a;
    real r = 3.0;
    real s = 2.0;
    a = min(r, s);
    print num2Str(a, 1, 2, 1, 1) + " is less than the other number.";
}
```

### power

Raises a real number to the power of another real number.

**Syntax**

```java
real power(real arg, real exponent)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg</td>
<td>The number to calculate the power of.</td>
</tr>
<tr>
<td>exponent</td>
<td>The number to raise the number that is specified by the <code>arg</code> parameter to.</td>
</tr>
</tbody>
</table>

**Return value**

The real number that is the number specified by the `arg` parameter to the power of the number specified by the `exponent` parameter.

**Remarks**

- `power(5.0, 2.0)` returns the value 25.0.
- `power(4.0, 0.5)` returns the value 2.0.

### round

Rounds a real number to the nearest multiple of another real number.

**Syntax**
real round(real _arg, real _decimals)

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_arg</td>
<td>The original number.</td>
</tr>
<tr>
<td>_decimals</td>
<td>The number that the value of the _arg parameter must be rounded to a multiple of.</td>
</tr>
</tbody>
</table>

**Return value**

The number that is a multiple of the value specified by the _decimals parameter and is closest to the value specified by the _arg parameter.

**Remarks**

To round a real number to a specified number of decimal places, use the decround function.

**Example**

- `round(123.45,5.00)` returns the value 125.00.
- `round(7.45,1.05)` returns the value 7.35.
- `round(23.9,5.0)` returns the value 25.00.
- `round(26.1,5.0)` returns the value 25.00.

---

real sin(real _arg)

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_arg</td>
<td>The number to calculate the sine for</td>
</tr>
</tbody>
</table>

**Return value**

The sine of the specified real number.

**Remarks**

The value of the _arg parameter must be in radians.

**Example**
static void sinExample(Args _arg)
{
    real angleDegrees = 15.0;
    real angleRadians;
    real pi = 3.14;
    real r;
    angleRadians = pi * angleDegrees / 180;
r = sin(angleRadians);
print "sin of a "
    + num2Str(angleDegrees, 2, 2, 1, 1)
    + " degree angle is "
    + num2Str(r, 2, 10, 1, 1);
pause;
}

sinh
Retrieves the hyperbolic sine of a real number.

Syntax

    real sinh(real _arg)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_arg</td>
<td>The number to calculate the hyperbolic sine for.</td>
</tr>
</tbody>
</table>

Return value
The hyperbolic sine of the specified real number.

Remarks
Values for the _arg parameter that are outside the -250 to 250 range cause the following run-time error: "Argument for trigonometric function out of range."

Example
The following example illustrates the sinh function.

static void sinhExample(Args _arg)
{
    real angleDegrees = 45.0;
    real angleRadians;
    real pi = 3.14;
    real r;
    angleRadians = pi * angleDegrees / 180;
r = sinh(angleRadians);
print "sinh of a "
    + num2Str(angleDegrees, 2, 2, 1, 1)
    + " degree angle is "
    + num2Str(r, 2, 10, 1, 1);
pause;
}

tan
Retrieves the tangent of a real number.

**Syntax**

```plaintext
real tan(real arg)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>arg</td>
<td>The real number to calculate the tangent for</td>
</tr>
</tbody>
</table>

**Return value**

The tangent of the specified real number.

**Remarks**

Values for the `arg` parameter that are outside the -250 to 250 range cause the following run-time error: "Argument for trigonometric function out of range."

**Example**

The following example illustrates the `tan` function.

```plaintext
static void tanExample(Args _arg)
{
    real r;
    r = tan(250);
    print strFmt("Tan of 250 is %1", r);
    pause;
}
```

---

**tanh**

Retrieves the hyperbolic tangent of a real number.

**Syntax**

```plaintext
real tanh(real _arg)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_arg</td>
<td>The number to calculate the hyperbolic tangent for</td>
</tr>
</tbody>
</table>

**Return value**

The hyperbolic tangent of the specified real number.

**Example**

The following example illustrates the `tanh` function.
static void tanhExample(Args _arg)
{
    real r;
    ;
    r = tanh(0.1);
    print "The hyperbolic tangent of angle 0.1 is "
    + num2Str(r, 2, 10, 1, 1);
    pause;
}

trunc
Truncates a real number by removing any decimal places.

Syntax

    real trunc(real _decimal)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_decimal</td>
<td>The number to truncate.</td>
</tr>
</tbody>
</table>

Return value

A number that is equivalent to the value of the _decimal parameter after the decimal places have been removed.

Remarks

This function always rounds numbers down to a complete integer.

Example

The following example truncates 2.7147 to 2.00.

    static void truncExample(Args _arg)
    {
        real r;
        ;
        r = trunc(2.7147);
        print strFmt("r = %1",  r);
        pause;
    }
This topic describes the reflection run-time functions.

**classIdGet**

Retrieves the numeric identifier (the class ID) of the class that the object that is initialized belongs to.

```c
int classIdGet(class object)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>The object to get the class ID for.</td>
</tr>
</tbody>
</table>

**Return value**
The class ID of the specified object.

**Example**

```c
static void classIdGetExample(Args _args)
{
    int i;
    WorkTimeCheck w;
    i = classIdGet(w);
    print "Class ID for object is " + int2Str(i);
}
```

**dimOf**

Retrieves the number of index elements that space has been allocated for in an X++ array.

```c
int dimOf(anytype object)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>The array to determine the dimension size of.</td>
</tr>
</tbody>
</table>

**Return value**
If the value of the `object` parameter is an array, the number of elements in the array; otherwise, 0 (zero).

**Remarks**
The `dimOf` function is intended for X++ arrays that are declared as the following X++ primitive types:

- boolean
- date
- int
- int64
- real
- utcDateTime

An example is `int iAmounts[6];`. Arrays of enumeration values and extended data types are also supported if they are ultimately based on one of the preceding primitive data types (such as `int`). The `dimOf` function doesn't accept arrays of all X++ primitive types. Here are the array types that the `dimOf` function doesn't accept:

- `str`
- `container`
- `anytype`
- Arrays of class objects
- Instances of the `Array` class

Example
static void JobDimOfArrays(Args _args)
{
    int iAmounts[20], iCounts[];
    ABCModel enumAbcModel[22]; // Enum
    ABCModelType extdtAbcModelType[24]; // Extended data type
    anytype anyThings[26];
    str sNames[28];
    Array myArrayObj; // Class

    info("Start of job.");
    info("--(Next, normal int array, dimOf() accepts it.)");
    info(int2Str(dimOf(iAmounts)));
    info("--(Next, normal enum array, dimOf() accepts it.)");
    info(int2Str(dimOf(enumAbcModel)));
    info("--(Next, normal extended data type array (based on enum), dimOf() accepts it.)");
    info(int2Str(dimOf(exdtAbcModelType)));
    info("--(Next, dynamic int array, dimension not yet set.)");
    info(int2Str(dimOf(iCounts)));
    info("--(Next, dynamic int array, after dimension established.)");

    iCounts[13] = 13;
    info(int2Str(dimOf(iCounts)));
    info("== == == == == (Next, array types that dimOf() does not support.)");
    info("--(Next, normal anytype array, dimOf() always returns 0.)");
    info(int2Str(dimOf(anyThings)));
    info("--(Next, an instance of class X++ Array, dimOf() always returns 0.)");

    myArrayObj = new Array(Types::Integer);
    myArrayObj.value(1, 501);
    info(int2Str(dimOf(myArrayObj)));
    info("--(Next, the lastIndex method provides size information about Array instances.)");
    info(int2Str(myArrayObj.lastIndex()));
    info("--(Next, normal str array, dimOf() does not accept it, job is halted.)");
    info(int2Str(dimOf(sNames)));
    info("End of job.");
}

************  Actual Infolog output
Message (11:10:06 am)
Start of job.
--(Next, normal int array, dimOf() accepts it.)
20
--(Next, normal enum array, dimOf() accepts it.)
22
--(Next, normal extended data type array (based on enum), dimOf() accepts it.)
24
--(Next, dynamic int array, dimension not yet set.)
0
--(Next, dynamic int array, after dimension established.)
16
== == == == (Next, array types that dimOf() does not support.)
--(Next, normal anytype array, dimOf() always returns 0.)
0
--(Next, an instance of class X++ Array, dimOf() always returns 0.)
0
--(Next, the lastIndex method provides size information about Array instances.)
1
--(Next, normal str array, dimOf() does not accept it, job is halted.)
Error executing code: Illegal operation on this type of array. (C)JobsJobDimOfArrays - line 41
************

***********  Pop-up error dialog box
"Internal error number 25 in script."
This error is caused by the code line...
info(int2Str(dimOf(iCounts)));
...before iCounts was assigned at any index.
***********/
fieldId2Name

Retrieves a string that represents the name of the field that is specified by a table ID number and a field ID number.

```c
str fieldId2Name(int tableid, int fieldid)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableid</td>
<td>The ID number of the table. <strong>Note:</strong> Use the <code>tableName2Id</code> function to specify the ID of a table.</td>
</tr>
<tr>
<td>fieldid</td>
<td>The ID number of the field.</td>
</tr>
</tbody>
</table>

**Return value**

The name of the field.

**Remarks**

To return a printable version of the field name, use the `fieldId2PName` function.

**Example**

The following example sets `fn` to the name of the field in the Customer (CustGroup) table that has a field ID of 7.

```c
static void fieldId2NameExample(Args _arg)
{
    str fn;
    fn = fieldId2Name(tableName2Id("Customer"),7);
}
```

fieldId2PName

Retrieves the printable name of the field that is specified by a table ID number and a field ID number.

```c
str fieldId2PName(int tableid, int fieldid)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableid</td>
<td>The ID number of the table. <strong>Note:</strong> Use the <code>tableName2Id</code> function to specify the ID of a table.</td>
</tr>
<tr>
<td>fieldid</td>
<td>The ID number of the field. <strong>Note:</strong> Use the <code>fieldName2Id</code> function to specify the ID of a field.</td>
</tr>
</tbody>
</table>

**Return value**

The name of the field.

**Example**
static void fieldId2PNameExample(Args _arg)
{
    str name;
    tableid _tableId;
    fieldid _fieldid;
    _tableId = tableName2Id("Address");
    _fieldId = fieldName2Id(_tableId, "Name");
    name = fieldId2PName(_tableId, _fieldId);
    print name;
}

fieldName2Id
Retrieves the field ID of the table field that is specified by a table ID number and a field ID number.

int fieldName2Id(int tableId, str fieldName)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableId</td>
<td>The ID number of the table. Note: Use the tableName2Id function to specify the ID of a table.</td>
</tr>
<tr>
<td>fieldName</td>
<td>The name of the field.</td>
</tr>
</tbody>
</table>

Return value
The ID of the field that is specified by the tableId and fieldName parameters.

Example

static void fieldName2IdExample(Args _arg)
{
    int id;
    id = fieldName2Id(tableName2Id("Address"), "Name");
    // Returns 6. Name is the 6th field in the Address table.
    print id;
}

indexId2Name
Retrieves the name of an index.

str indexId2Name(int tableId, int indexId)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableId</td>
<td>The ID of the table that the index belongs to.</td>
</tr>
<tr>
<td>indexId</td>
<td>The ID of the index.</td>
</tr>
</tbody>
</table>
Return value
The name of the index.

Example

```java
static void indexId2NameExample(Args _arg)
{
    str s;
    tableid id;
    indexid idx;

    id  = tableName2Id("Address");
    idx = indexName2Id(id, "AddrIdx");
    s = indexId2Name(id, idx);
    print "The result of calling indexId2Name is " + s;
}
```

indexName2Id
Retrieves the ID of an index.

```java
int indexName2Id(int tableid, str indexname)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>tableid</td>
<td>The ID of the table that the index belongs to.</td>
</tr>
<tr>
<td>indexname</td>
<td>The name of the index.</td>
</tr>
</tbody>
</table>

Return value
The ID of the index.

Example

```java
static void indexName2IdExample(Args _arg)
{
    indexid idx;
    tableid id;

    id  = tableName2Id("Address");
    idx = indexName2Id(id, "AddrIdx");
    print "Index ID for index name AddrIdx of table Address is " + int2Str(idx);
}
```

tableId2Name
Retrieves a string that contains the name of a table.

```java
str tableId2Name(int _tableid)
```

Parameters
<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_tableid</td>
<td>The ID of the table.</td>
</tr>
</tbody>
</table>

**Return value**
The name of the table.

**Example**

```java
static void tableId2NameExample(Args _arg)
{
    str s;
    tableId id;

    // Get the ID for table name Address.
    id = tableName2Id("Address");
    print "ID for table name Address is " + int2Str(id);

    // Get the name from the table ID.
    s = tableId2Name(id);
    print "Name for table ID " + int2Str(id) + " is " + s;

    // Get the printable name from the table ID.
    s = tableId2PName(id);
    print "Printable name for table ID " + int2Str(id) + " is " + s;
}
```

**tableId2PName**
Retrieves a string that contains the printable name (the label) of a table.

```java
str tableId2PName(int _fieldid)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_fieldid</td>
<td>The ID of the table.</td>
</tr>
</tbody>
</table>

**Return value**
The label of the table.

**Example**
```java
static void tableId2NameExample(Args _arg)
{
    str s;
    tableid id;

    // Get the ID for table name Address.
    id = tableName2Id("Address");
    print "ID for table name Address is " + int2Str(id);

    // Get the name from the table ID.
    s = tableId2Name(id);
    print "Name for table ID " + int2Str(id) + " is " + s;

    // Get the printable name from the table ID.
    s = tableId2PName(id);
    print "Printable name for table ID " + int2Str(id) + " is " + s;
}
```

tableName2Id
Retrieves the ID of a table.

```java
int tableName2Id(str _name)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_name</td>
<td>The name of the table.</td>
</tr>
</tbody>
</table>

Return value
The ID of the table.

Example
```java
static void tableName2IdExample(Args _arg)
{
    str s;
    tableid id;

    // Get the ID for the Address table name.
    id = tableName2Id("Address");
    print "ID for the Address table name is " + int2Str(id);

    // Get the name from the table ID.
    s = tableId2Name(id);
    print "Name for table ID " + int2Str(id) + " is " + s;

    // Get the printable name from the table ID.
    s = tableId2PName(id);
    print "Printable name for table ID " + int2Str(id) + " is " + s;
}
```

typeOf
Retrieves the type of an element.
enum typeof(anytype _object)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_object</td>
<td>The element to return the type for</td>
</tr>
</tbody>
</table>

Return value

A Types system enumeration value.

Example

The following example tests whether the first element in a container, c, is another container that contains a single integer.

```c
if(typeof(conpeek(c, 1)) != Types::Container ||
    conlen(conpeek(c, 1)) != 1 ||
    typeof(conpeek(conpeek(c, 1), 1)) != Types::Integer)
{
    // More code.
}
```
This topic describes the session run-time functions.

**curExt**

Retrieves the extension that is used for the current company.

```csharp
str curExt()
```

**Return value**

The extension for the current company.

**Example**

```csharp
static void curExtExample(Args _arg)
{
    str s;
    // Sets s to the extension of the current company.
    s = curExt();
    print "Current extension is " + s;
}
```

**curUserId**

Retrieves the nonnumeric ID that represents the current user.

```csharp
str curUserId()
```

**Return value**

The nonnumeric ID that represents the current user.

**Example**

```csharp
static void curUserIdExample(Args _arg)
{
    str s;
    s = curUserId();
    print "Current user ID is " + s;
}
```

**funcName**

Retrieves a string that contains the current function context.

```csharp
str funcName()
```

**Return value**
The name of the method that is executing this method.

Remarks

If execution is currently within the member of a table or class, the name of the method is prefixed with the name of that table or class.

Example

```java
static void funcNameExample(Args _arg)
{
    print "Current function context is " + funcName();
}
```

**getcurrentPartition**

Retrieves the short name of the current partition.

```java
str getCurrentPartition()
```

Return value

The short name of the current partition.

Remarks

The maximum length of the data partition name that is returned is eight characters.

Example

The following code example shows calls to, and output from, the `getCurrentPartition` function of the X++ language, and related functions or methods.

```java
static public void Main(Args _args)  // X++ method.
{
    int64 iPartition;
    str sPartition;
    SelectableDataArea oSelectableDataArea;  // System ExDT.
    iPartition = getCurrentPartitionRecId();
    sPartition = getcurrentpartition();
    oSelectableDataArea = Global::getCompany( tableNum(BankAccountTable) );
    Global::info( strFmt( "getCurrentPartitionRecId =%1 , getCurrentPartition =%2 , getCompany =%3",
        iPartition, sPartition, oSelectableDataArea) );
}
/**** Pasted from Infolog window:
Message_@SYS14327 (03:42:38 pm)
gGetCurrentPartitionRecId =5637144576 , getCurrentPartition =initial , getCompany =ceu
****/```

**getcurrentPartitionRecId**

Retrieves the RecId field of the current partition.

```java
int64 getCurrentPartitionRecId()
```

Return value

The RecId field of the current data partition.
Remarks
To see a code example that relies on the `getCurrentPartitionRecId` function, see How to: Include a Filter for Partition in Direct Transact-SQL.

Example
The following code example shows calls to, and output from, the `getCurrentPartitionRecId` function of the X++ language, and related functions or methods.

```csharp
static public void Main(Args _args)  // X++ method.
{
    int64 iPartition;
    str sPartition;
    SelectableDataArea oSelectableDataArea;  // System ExDT.
    iPartition = getCurrentPartitionRecId();
    sPartition = getcurrentpartition();
    oSelectableDataArea = Global::getCompany( tableNum(BankAccountTable) );
    Global::info( strFmt("getCurrentPartitionRecId =%1 , getCurrentPartition =%2 , getCompany =%3",
                        iPartition, sPartition, oSelectableDataArea) );
}

**** Pasted from Infolog window:
Message_@SYS14327 (03:42:38 pm)
ggetCurrentPartitionRecId =5637144576 , getCurrentPartition =initial , getCompany =ceu
****
```

getPrefix
Retrieves the current execution prefix after successive calls to the `setPrefix` function.

```csharp
str getPrefix()
```

Return value
The current execution prefix.

Remarks
The prefix mechanism makes it more straightforward to write precise error messages about the transactions that an application performs. Because a hierarchical display is created in the Infolog, it can be easier to determine where each error came from.

Example
```csharp
static void getPrefixExample(Args _arg)
{
    setPrefix("Prefix");
    setPrefix("Another prefix");
    print getPrefix();
}
```

sessionId
Retrieves the session number of the current session.

```csharp
int sessionId()
```

Return value
The numeric ID of the current session.

**Remarks**
A session number is assigned when the client is started and connects to Application Object Server (AOS). Every call of this function during the life of the client returns the same integer value. The returned value is compatible with the `SessionID` extended data type. The `contains` methods return information about individual user sessions.

**Example**

```csharp
static void sessionIdExample(Args _arg)
{
    int session;
    session = sessionId();
    print "This session ID is number " + int2Str(session);
}
```

**prmIsDefault**
Determines whether the specified parameter for the current method has the default value.

```csharp
int prmIsDefault(anytype argument)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Argument</td>
<td>The parameter to test.</td>
</tr>
</tbody>
</table>

**Return value**

1 if the default value for the parameter was used; otherwise, 0 (zero).

**Example**

```csharp
static void prmIsDefaultExample(Args _arg)
{
    void fn(boolean b = true, int j = 42)
    {
        if (prmIsDefault(b) == 1)
        {
            print "First parameter is using the default value."
        }
        else
        {
            print "First parameter is not using the default value."
        }
    }
    fn();
    fn(false);
}
```

**runAs**
Enables the caller to run an X++ method in the security context of another user. This function is most often used with batch processing.
container runAs(
    str userId,
    int classId,
    str staticMethodName
    [, 
    container params,
    str company,
    str language,
    str partition ])

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>userId</td>
<td>The user to impersonate.</td>
</tr>
<tr>
<td>classId</td>
<td>The class to invoke in the impersonated session.</td>
</tr>
<tr>
<td>staticMethodName</td>
<td>The class method to invoke in the new user context.</td>
</tr>
<tr>
<td>params</td>
<td>The parameters to pass to the method; optional.</td>
</tr>
<tr>
<td>company</td>
<td>The company that is selected for the impersonated session; optional.</td>
</tr>
<tr>
<td>language</td>
<td>The language that is selected for the impersonated session; optional.</td>
</tr>
<tr>
<td>partition</td>
<td>The partition key of the type that is returned by the getCurrentPartition function; optional.</td>
</tr>
</tbody>
</table>

**Return value**

A container that holds the return value or values of the method that is called by the `runAs` function, if any values were returned.

**Remarks**

This function makes it possible to run code as another user. This capability presents a security threat. Therefore, this function runs under Code Access Security. Calls to this function on the server require permission from the `RunAsPermission` class. Each use of this application programming interface (API) should be threat-modeled. If a security vulnerability is discovered, validate input to this API. The debugger might ignore breakpoints that are located in a method that is called by using the `runAs` function. X++ code that is executed by the `runAs` function must run as Microsoft .NET Framework Common Intermediate Language (CIL). If CIL hasn't been generated for the target static method, an error message indicates that the method isn't found. The `PartitionKey` system type is the exact type of the `partition` parameter. `PartitionKey` is a string that has a maximum length of eight characters.

**Example**

The following example calls the `runDueDateEventsForUser` method in the `EventJobDueDate` class. The code runs in the security context of a user. Run this code by applying it to a method in a new class.
server static public void Main(Args _args)
{
    RunAsPermission perm;
    UserId runAsUser;
    SysUserInfo userInfo;
    userInfo = SysUserInfo::find();
    runAsUser = userInfo.Id;
    perm = new RunAsPermission(runAsUser);
    perm.assert();
    runAs(runAsUser, classnum(EventJobDueDate), "runDueDateEventsForUser");
    CodeAccessPermission::revertAssert();
}

setPrefix

Sets the prefix for the current execution scope.

int setPrefix(str _prefix)

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_prefix</td>
<td>The prefix for the current execution scope.</td>
</tr>
</tbody>
</table>

Return value

0 if the prefix was set successfully.

Remarks

The complete prefix for the execution can be fetched by using the getPrefix function. When the scope is left, the prefix is automatically reset to the previous level. The prefix mechanism makes it more straightforward to write precise error messages about the transactions that an application performs. For example, the AA method calls the BB method, and each method calls the setPrefix function. Messages that the BB method writes to the Infolog appear nested in a hierarchy. When the BB method ends, and control returns to the AA method, the prefix that was set by the BB method isn't attached to subsequent messages.

Example

static void setPrefixExample(Args _arg)
{
    int i;
    i = setPrefix("Prefix");
    print i;
}
This topic describes the string run-time functions.

**match**

Searches for a string or expression in another string.

```c
int match(str pattern, str text)
```

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>pattern</td>
<td>The string or expression to search for.</td>
</tr>
<tr>
<td>text</td>
<td>The string to search.</td>
</tr>
</tbody>
</table>

**Return value**

1 if the pattern is located in the string; otherwise, 0 (zero).

**Remarks**

The search is case-insensitive. The following special characters can be used to create the pattern for the `pattern` parameter.

- \: A backslash (\) nullifies, or escapes, the special treatment of special characters, so that a special character can be matched like a normal letter. A pair of backslashes is translated into one non-special backslash. Examples:
  - `match("ab\$cd","ab$cd")`; returns 0.
  - `match("ab\$cd","ab$cd")`; returns 0. The backslash isn't escaped.
  - `match("ab\\$cd","ab$cd")`; returns 1. The backslash and dollar sign are escaped.

- < or ^: A left angle bracket (<) or a circumflex (^) at the start of an expression is used to match the start of a line. Examples:
  - `match("<abc","abcdef")`; returns 1.
  - `match("<abc","defabc")`; returns 0.
  - `match("^abc","abcdef")`; returns 1.
  - `match("^abc","defabc")`; returns 0.

- > or $: A right angle bracket (>) or a dollar sign ($) at the end of the expression is used to match the end of a line. Examples:
  - `match("abc>","abcdef")`; returns 0.
  - `match("abc>","defabc")`; returns 1.

- ? or .: A question mark (?) or a period (.) matches any one character in the same position. Examples:
  - `match("abc.def","abc#def")`; returns 1.
  - `match("colou?r","colouXr")`; returns 1.
A colon (:) specifies a group of characters to match, as indicated by the character that immediately follows.

- `:a`: Sets the match to letters. Examples:
  - `match("ab:acd","ab#cd")`; returns 0.
  - `match("ab:acd","abxyzcd")`; returns 0.
  - `match("ab:acd","abbcd")`; returns 1.

- `:d`: Sets the match to numeric characters. Examples:
  - `match("ab:dcd","ab3cd")`; returns 1.
  - `match("ab:dcd","ab123cd")`; returns 0.
  - `match("ab:dcd","abcd")`; returns 0.

- `:n`: Sets the match to alphanumeric characters. Examples:
  - `match("ab:ncd","ab%cd")`; returns 0.
  - `match("ab:ncd","ab9cd")`; returns 1.
  - `match("ab:ncd","abXcd")`; returns 1.

- `:SPACE`: SPACE is the space character (" "). Sets the match to blanks, tabulations, and control characters such as Enter (new line). Examples:
  - `match("ab: cd","ab cd")`; returns 1.
  - `match("ab: cd","ab\ncd")`; returns 1.
  - `match("ab: cd","ab\tcd")`; returns 1.
  - `match("ab: cd","ab  cd")`; returns 0. Only the first space is matched.

- `*:` An expression that is followed by an asterisk ("*") requires a match for zero, one, or more occurrences of the preceding expression. Examples:
  - `match("abc*d","abd")`; returns 1.
  - `match("abc+d","abcd")`; returns 1.
  - `match("abc*d","abcccd")`; returns 1.
  - `match("abc+d","abxd")`; returns 0.

- `+:` An expression that is followed by a plus sign (+) requires a match for one or more occurrences of the preceding expression. Examples:
  - `match("abc+d","abd")`; returns 0.
  - `match("abc+d","abcd")`; returns 1
  - `match("abc+d","abcccd")`; returns 1.
  - `match("abc+d","abxd")`; returns 0.

- `-:` An expression that is followed by a minus sign (-) requires a match for zero or one occurrence of the preceding expression. In other words, the preceding expression is optional. Examples:
  - `match("colo\-r","color")`; returns 1.
  - `match("colo\-r","colour")`; returns 1.

- `[]`: Matches a single character with any character that is enclosed in the brackets. A range of characters can be specified by two characters that are separated by a minus sign (-). For example, `[a-z]` matches all letters between a and z, `[0-9]` matches a digit, and `[0-9a-f]` matches a hexadecimal digit. Examples:
  - `match("[abc]","apple")`; returns 1, because it matches the a in "apple."
  - `match("[abc]","kiwi")`; returns 0, because "kiwi" doesn't contain an a, b, or c.
  - `match("gr[ae]y","grey")`; returns 1. This expression also matches "gray."
  - `match("gr[ae]y","graey")`; returns 0, because only one character between "gr" and "y" is matched.
strAlpha
Copies only the alphanumerical characters from a string.

```c
str strAlpha(str _text)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_text</td>
<td>The string to copy from.</td>
</tr>
</tbody>
</table>

Return value
A new string that contains all the alphanumerical characters from the specified string.

Remarks
For example, `strAlpha("2+2=5 is this correct?")` returns the string `225isthiscorrect`.

Example

```c
static void strAlphaExample(Args _arg)
{
    str s;
    s = strAlpha("?a*bc123.");
    print s;
    pause;
}
```

strCmp
Compares two text strings.

```c
int strCmp(str text1, str text2)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>text1</td>
<td>The first string.</td>
</tr>
<tr>
<td>text2</td>
<td>The second string.</td>
</tr>
</tbody>
</table>

Return value
0 if the two strings are identical, 1 if the first string sorts earlier, or -1 if the second string sorts earlier.

Remarks

- `[`: If the first character in the text that is enclosed in brackets is a circumflex (^), the expression matches all characters except the characters that are enclosed in the brackets. Examples:
  - `match("[^bc]at","bat");` returns 0.
  - `match("[^bc]at","hat");` returns 1.
  - `match("[^abc]","bat");` returns 1. Anything except a, b, or c is matched. Therefore, the t is matched.
The comparison performed by this method is case-sensitive.

```c
print strCmp("abc", "abc"); //Returns the value 0.
print strCmp("abc", "ABC"); //Returns the value 1.
print strCmp("aaa", "bbb"); //Returns the value -1.
print strCmp("ccc", "bbb"); //Returns the value 1.
```

### strColSeq

Converts all uppercase characters to lowercase characters, and converts all characters that have accents to the corresponding unaccented lowercase characters.

```c
str strColSeq(str text)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>The string to copy and convert characters from.</td>
</tr>
</tbody>
</table>

**Return value**

The converted text string.

**Remarks**

The `strColSeq` function exists for backward-compatibility purposes. This function supports only the mapping for the following Western European characters:

- AâáàâãäÀÁÂÃÄBCçÇDEééèéêëÈÉÊËFGHIìíîïÌÍÎÏJKLMNñÑOòóôõöÒÓÔÕÖPQRSTUùúûúÜÚÜWXYYÝZææéÆØÅ
- aaaaaaaaabccddeeeeeeefghiiiiiiijklmnnnoooooonopqrstuvwxyz~Ç~Ç

For Unicode-compliant functionality, use the Win32 LMapString application programming interface (API) via the DLL and DLLFunc classes.

**Example**

The following example prints `abcdeabcde`.

```c
static void strColSeqExample(Args _arg)
{
    
    print strColSeq(""),
    print strColSeq(""),
    pause;
}
```

### strDel

Creates a copy of a string, from which the specified substring is removed.

```c
str strDel(str _text, int _position, int _number)
```

**Parameters**
### strDel

The string to copy from.

The position at which to begin ignoring characters during the copy operation.

The number of characters to ignore. A minus sign in front of the _number parameter indicates that _number–1 characters before the character at _position should be removed together with the character at _position.

**Return value**

The remaining characters that are copied from the string.

**Remarks**

The `strDel` function is complementary to the `substr` function.

```
strDel("ABCDEFGH",2,3); //Returns the string "AEFGH".
strDel("ABCDEFGH",4,3); //Returns the string "ABCGH".
```

### strFind

Searches a string for the first occurrence of one of the specified characters.

```
int strFind(str _text, str _characters, int _position, int _number)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_text</td>
<td>The string to search.</td>
</tr>
<tr>
<td>_characters</td>
<td>The characters to search for.</td>
</tr>
<tr>
<td>_position</td>
<td>The position in the string where the search begins.</td>
</tr>
<tr>
<td>_number</td>
<td>A signed number that indicates the direction of the search and how many positions to search in the string.</td>
</tr>
</tbody>
</table>

**Return value**

The value of the position of the first occurrence of one of the specified characters, or 0 when none found.

**Remarks**

To search from the beginning of the string to the end, use 1 as the value of the _position parameter. If the value of the _number parameter is negative, the system searches the number of characters backward from the specified position. The search isn't case-sensitive. Here is an example.

```
strFind("ABCDEFGHIJ","KHD",1,10); //Returns the value 4 (the position where "D" was found).
strFind("ABCDEFGHIJ","KHD",10,-10); //Returns the value 8 (the position where "H" was found).
```

The `strFind` function is complementary to the `strNFind` function.
strFmt

Formats the specified string and substitutes any occurrences of n with the nth argument.

\[ \text{str strFmt(str _string, ...)} \]

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_string</td>
<td>The strings to format.</td>
</tr>
</tbody>
</table>

**Return value**

The formatted string.

**Remarks**

If an argument isn't provided for a parameter, the parameter will be returned as "%n" in the string. The string conversion of values of the `real` type is limited to two decimal places. Values are rounded, not truncated. The `System.String::Format` method from the Microsoft .NET Framework can be used to gain additional functionality, as shown in the example.

**Example**

```csharp
static void strFmtExampleJob(Args _arg)
{
    System.Double sysDouble;
    real r = 8.3456789;
    int  i = 42;
    utcDateTime utc = str2DateTime("2008-01-16 13:44:55",321); // 321 == YMD.
    str  s;

    s = strFmt("real = %1, int = %2, utcDateTime = %3, [%4]", r, i, utc);
    info("X1:  " + s);
    //
    sysDouble = r;
    s = System.String::Format("{0:##.####}", sysDouble);
    info("N1:  " + s);
    //
    s = System.String::Format("{0,6:C}", sysDouble); // $
    info("N2:  " + s);
    //********** Actual Infolog output
    Message (02:16:05 pm)
    X1:  real = 8.35, int = 42, utcDateTime = 1/16/2008 01:44:55 pm, [%4]
    N1:  8.3457
    N2:  $8.35
    **********/
}
```

strIns

Builds a string by inserting one string into another.

\[ \text{str strIns(str _text1, str _text2, int _position)} \]

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
**PARAMETER** | **DESCRIPTION**
--- | ---
_text1 | The string to insert the other string into.
_text2 | The string to insert into the other string.
_position | The position where the first character of the _text2 parameter should occur in the output string.

**Return value**
The combined text string.

**Remarks**
The **strIns** function is complementary to the **strDel** function. If the value of the _position parameter is more than the length of the original string, the string to insert is appended to the end of the original string.

```c
strIns("ABFGH","CDE",3); //Returns the string "ABCDEFGH".
strIns("ABCD","EFGH",10); //Returns the string "ABCDEFGH".
```

**strKeep**
Builds a string by using only the characters from the first input string that the second input string specifies should be kept.

```c
str strKeep(str _text1, str _text2)
```

**Parameters**

| PARAMETER | DESCRIPTION |
--- | --- |
_text1 | The string that contains the characters that can be used to build an output string. |
_text2 | The string that specifies which characters to keep for the output string. |

**Return value**
A string of the characters that are kept.

**Remarks**
The **strKeep** function is complementary to the **strRem** function.

```c
strKeep("ABBCDEFGHB","BCD"); //Returns the string "BBCDB".
strKeep("abcZcba","bc") //Returns the string "bccb".
```

The **strKeep** function is complementary to the **strRem** function.

**strLen**
Calculates the length of the specified string.

```c
int strLen(str text)
```
### strLen

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>The string to calculate the length of</td>
</tr>
</tbody>
</table>

**Return value**

The length of the specified string.

**Remarks**

```plaintext
strLen("ABC"); //Returns the value 3.
strLen("ABCDEFGHIJ"); //Returns the value 10.
```

### strLine

Retrieves a single line from a string that spans multiple lines.

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string that might span multiple lines.</td>
</tr>
<tr>
<td>count</td>
<td>The offset of the line to return.</td>
</tr>
</tbody>
</table>

**Return value**

A copied line of the string that is specified by the `string` parameter.

**Remarks**

The first line of the string has an offset of 0. You can assign multiple lines to one string by embedding the `\n` or `\r\n` characters in the string. Additionally, you can use the at sign (@) immediately before the opening quotation mark and use the Enter key to spread parts of the string value over multiple lines in the X++ code editor.

**Example**

```plaintext
str mytxt = "first-line\nsecond-line\nlast-line";
// Prints "second-line".
print strLine(mytxt,1);
// Prints "last-line".
print strLine(mytxt,2);
```

### strLTrim

Removes leading blanks from a text string.

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td></td>
</tr>
</tbody>
</table>
### strLTrim

The string equivalent for the text that leading blanks have been removed from.

**Remarks**
The `strLTrim` function is complementary to the `strRTrim` function.

**Example**

```c
// Returns the text string "ABC-DEFG".
strLTrim("   ABC-DEFG");
```

### strLwr

Converts all letters in the specified string to lowercase.

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_text</td>
<td>The string to convert to lowercase.</td>
</tr>
</tbody>
</table>

**Return value**

A copy of the specified string that contains only lowercase letter.

**Remarks**
The `strLwr` function is complementary to the `strUpr` function. The `strLwr` function uses the `LCMapString` function in the Win32 API.

**Example**

```c
static void strLwrExample(Args _args)
{
    // Returns the text string "abcd55efghij".
    print strLwr("Abcdd55EFGHIJ");
    pause;
}
```

### strNFind

Searches part of a text string for the first occurrence of a character that isn't included in the specified list of characters.

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_text</td>
<td>The string equivalent for the text that leading blanks have been removed from.</td>
</tr>
<tr>
<td>_characters</td>
<td>The string equivalent for the text that trailing blanks have been removed from.</td>
</tr>
<tr>
<td>_position</td>
<td>The position to start searching from.</td>
</tr>
<tr>
<td>_number</td>
<td>The number of characters to search for.</td>
</tr>
</tbody>
</table>

The `strNFind` function is complementary to the `strRFind` function.
### strNFind

The position of the first occurrence of a character that isn't specified by the `_characters` parameter, or 0 when none found.

#### Remarks

The search isn't case-sensitive. To search from the beginning of the string to the end, use a value of 1 for the `_position` parameter. If a minus sign precedes the value of the `_number` parameter, the characters will be searched in reverse order, starting from the position that is specified by the `_position` parameter.

```plaintext
strNFind("ABCDEFGHIJKLMNOPQRSTUVWXYZ", "ABCDHIJ", 1, 10); // Returns the value 5 (the position of "E");
strNFind("CDEFGHIJ", "CDEFGI", 10, -10); // Returns the value 6 (the position of "H");
strNFind("abcdef", "def", 3, 2) // Returns the value 4.
```

The `strNFind` function is complementary to the `strFind` function.

### strPoke

Overwrites part of a string with another string.

```plaintext
strPoke(str _text1, str _text2, int _position)
```

#### Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_text1</td>
<td>The original string.</td>
</tr>
<tr>
<td>_text2</td>
<td>The string to replace part of the original string with.</td>
</tr>
<tr>
<td>_position</td>
<td>The position of the original string at which to begin replacing the characters.</td>
</tr>
</tbody>
</table>

#### Return value

The new string.

#### Remarks

The new string can be longer than the original string. However, if the value of the `_position` parameter is more than the length of the string, the original string is returned without replacements.
strPoke("12345678","AAA",3);  //Returns the string "12AAA678".
strPoke("abcde","4567",4);  //Returns the string "abc4567".
strPoke("abcde", "4567", "10");  //Returns the string "abcde".

**strPrompt**

Appends a string with the specified number of period characters, followed by a colon and space character.

```c
str strPrompt(str _string, _int len)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_string</td>
<td>The original string.</td>
</tr>
<tr>
<td>_len</td>
<td>The desired final length of the string.</td>
</tr>
</tbody>
</table>

**Return value**

A string that looks like a prompt for user input.

**Remarks**

In atypical cases, where the value of the _len parameter is only slightly more than the length of the original string, the highest precedence is given to adding the trailing space. Next, precedence is given to the colon. The lowest precedence is given to the periods. Negative values for the _len parameter return the input string appended with a trailing space.

```c
strPrompt("ab",-1);  //Returns "ab ".
strPrompt("ab",3);   //Returns "ab ".
strPrompt("ab",4);   //Returns "ab: ".
strPrompt("ab",5);   //Returns "ab:. ".
strPrompt("ab",6);   //Returns "ab... ".
```

**Example**

```c
static void JobStrPromptDemo(Args _args)
{
    // Printed string is "[abc..:]
    print "[", strPrompt("abc", 7), "]";
    pause;
}
```

**strRem**

Removes the characters that are specified in one string from another string.

```c
str strRem(str text1, str text2)
```

**Parameters**
### strRem

The remaining content of the original string.

**Remarks**

This function is case-sensitive.

```c
strRem("abcd_abcd","Bc"); //Returns the string "abd_abd".
strRem("ABCDEFGABCDEFG","ACEG"); //Returns the string "BDFBDF".
```

This function is complementary to the `strKeep` function.

### strRep

Repeats a string of characters.

```c
str strRep(str _text, str _number)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_text</td>
<td>The string to repeat.</td>
</tr>
<tr>
<td>_number</td>
<td>The number of times to repeat the string.</td>
</tr>
</tbody>
</table>

**Return value**

A new string that contains the contents of the original string that are repeated the specified number of times.

**Example**

The following example prints the text string **ABABABABABAB**.

```c
static void strRepExample(Args _arg)
{
    str strL;
    strL = strRep("AB",6);
    print strL;
    pause;
}
```

### strRTrim

Removes the trailing space characters from the end of a string.

```c
str strRTrim(str _text)
```
### Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_text</td>
<td>The string to remove the trailing space characters from.</td>
</tr>
</tbody>
</table>

### Return value

A copy of the specified string that doesn't include trailing space characters.

### Remarks

The `strRTrim` function is complementary to the `strLTrim` function.

### strScan

Searches a text string for an occurrence of another string.

```plaintext
int strScan(str _text1, str _text2, int _position, int _number)
```

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_text1</td>
<td>The string to search in.</td>
</tr>
<tr>
<td>_text2</td>
<td>The string to find.</td>
</tr>
<tr>
<td>_position</td>
<td>The first position in the _text1 parameter at which to perform a comparison.</td>
</tr>
<tr>
<td>_number</td>
<td>The number of positions in the _text1 parameter to retry the comparison for. If a minus sign precedes the _number parameter, the system searches the number of characters in reverse order from the specified position.</td>
</tr>
</tbody>
</table>

### Return value

The position at which the specified string was found in the string; otherwise, 0 (zero).

### Remarks

The comparisons aren't case-sensitive. Values for the _position parameter that are less than 1 are treated as 1. The direction of the scan is controlled by the sign that is specified in the _number parameter. A positive sign indicates that each successive comparison will start one position closer to the end of the string. A negative sign indicates that each comparison will start one position closer to the start of the string.

```plaintext
strScan("ABCDEFGHIJ","DEF",1,10); //Returns the value 4.
strScan("ABCDEFGHIJ","CDE",10,-10); //Returns the value 3.
```

### strUpr

Converts all the letters in a string to uppercase.
```csharp
str strUpr(str _text)
```

### Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_text</td>
<td>The string to convert to uppercase letters.</td>
</tr>
</tbody>
</table>

### Return value

A copy of the specified string that contains only lowercase letters.

### Remarks

The `strUpr` function is complementary to the `strLwr` function. The `strUpr` function uses the `LCMapString()` function in the Win32 API.

### Example

The following example will print `ABCDD55EFGHIJ`.

```csharp
static void strUprExample(Args _args)
{
    print strUpr("Abcdd55EFGhiJ");
    pause;
}
```

---

### subStr

Retrieves part of a string.

```csharp
str subStr(str _text, int _position, int _number)
```

### Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_text</td>
<td>The original string.</td>
</tr>
<tr>
<td>_position</td>
<td>The position in the original string where the part to retrieve begins.</td>
</tr>
<tr>
<td>_number</td>
<td>A signed integer that indicates the direction and number of positions to retrieve from the original string. If a minus sign precedes <code>_number</code>, the system selects the substring backward from the specified position.</td>
</tr>
</tbody>
</table>

### Return value

A substring of the original string.

### Remarks

If a minus sign precedes the value of the `_number` parameter, the substring will be selected backward from the specified position.
subStr("ABCDEFGHIJ",3,5); //Returns the string "CDEFG".
subStr("ABCDEFGHIJ",7,-4); //Returns the string "DEFG".
subStr("abcdef",2,99) //Returns the string "bcdef".
subStr("abcdef",2,3) //Returns the string "bcd".
subStr("abcdef",2,-3); //Returns the string "ab".
This topic contains the documentation available for the System classes.

**NOTE**
This topic is not a complete list of the System table members. You can find a complete list of tables and their members in the Application Explorer.

## Common

The Common table is the base class for all tables. It does not contain any data. It is primarily used in X++ code to refer to any table in a polymorphic way.

### Methods

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>aosValidateDelete</td>
<td>Validates on the server that the specified record can be deleted from a table.</td>
</tr>
<tr>
<td>aosValidateInsert</td>
<td>Validates on the server that the specified record can be inserted.</td>
</tr>
<tr>
<td>aosValidateRead</td>
<td>Validates on the server that the specified record can be read.</td>
</tr>
<tr>
<td>aosValidateUpdate</td>
<td>Validates on the server that the specified record can be updated.</td>
</tr>
<tr>
<td>buf2con</td>
<td>Packs the table buffers of an xRecord instance into an X++ container.</td>
</tr>
<tr>
<td>canSubmitToWorkflow</td>
<td>Indicates whether submission to workflow is possible.</td>
</tr>
<tr>
<td>caption</td>
<td>Gets and sets the caption property of a table.</td>
</tr>
<tr>
<td>checkInvalidFieldAccess</td>
<td>Gets and sets invalid field access.</td>
</tr>
<tr>
<td>checkRecord</td>
<td>Gets and sets the property that indicates whether to check mandatory fields.</td>
</tr>
<tr>
<td>checkRestrictedDeleteActions</td>
<td>Gets and sets the property that indicates whether a record can be deleted.</td>
</tr>
<tr>
<td>clear</td>
<td>Removes all rows from the table buffer.</td>
</tr>
<tr>
<td>company</td>
<td>Gets and sets the property that indicates a legal entity for the record.</td>
</tr>
<tr>
<td>METHOD</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>con2buf</td>
<td>Unpacks a container into the table buffers.</td>
</tr>
<tr>
<td>concurrencyModel</td>
<td>Gets and sets the default concurrency model to use to update records.</td>
</tr>
<tr>
<td>context</td>
<td>Gets and sets the context property.</td>
</tr>
<tr>
<td>data</td>
<td>Retrieves a row from the table.</td>
</tr>
<tr>
<td>dataSource</td>
<td>Retrieves the data source of the table.</td>
</tr>
<tr>
<td>dbOpInTransaction</td>
<td>Makes sure that database operations are correctly closed if they fail.</td>
</tr>
<tr>
<td>defaultField</td>
<td>Populates default values in a field in the table.</td>
</tr>
<tr>
<td>defaultRow</td>
<td>Populates default values in fields in the table in the non-interactive case.</td>
</tr>
<tr>
<td>delete</td>
<td>Deletes the current record from the table.</td>
</tr>
<tr>
<td>disableCache</td>
<td>Gets and sets the property that indicates whether caching is disabled.</td>
</tr>
<tr>
<td>dispose</td>
<td>Releases resources that are used by the xRecord object.</td>
</tr>
<tr>
<td>doClear</td>
<td>Removes all rows from the table buffer and bypasses any additional logic in the clear method of the table.</td>
</tr>
<tr>
<td>doDelete</td>
<td>Deletes the current record from the table and bypasses any additional logic in the delete method of the table.</td>
</tr>
<tr>
<td>doinsert</td>
<td>Inserts the record into the table and bypasses any additional logic in the insert method of the table.</td>
</tr>
<tr>
<td>doUpdate</td>
<td>Updates the current record and bypasses any additional logic in the update method of the table.</td>
</tr>
<tr>
<td>doValidateDelete</td>
<td>Performs the action to validate that a record can be deleted.</td>
</tr>
<tr>
<td>equal</td>
<td>Determines whether the specified object is equal to the current one.</td>
</tr>
<tr>
<td>fieldAccessRight</td>
<td>Returns the field access right.</td>
</tr>
<tr>
<td>fieldBufferAccessRight</td>
<td>Returns the field access right for the current record.</td>
</tr>
<tr>
<td>fieldState</td>
<td>Sets or returns the state of a field in the table buffer.</td>
</tr>
<tr>
<td>getAllowRedefault</td>
<td>Returns the list of fields that are allowed to re-default.</td>
</tr>
<tr>
<td>getDefaultValueingDependencies</td>
<td>Returns the container that holds defaulting dependencies.</td>
</tr>
<tr>
<td>METHOD</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>getExtension</td>
<td>Returns the table extension.</td>
</tr>
<tr>
<td>getFieldValue</td>
<td>Gets the value of the specified field from a table buffer.</td>
</tr>
<tr>
<td>getInstanceRelationType</td>
<td>Returns the table name that corresponds to the InstanceRelationType ID.</td>
</tr>
<tr>
<td>getPhysicalTableName</td>
<td>Return the physical table name, which, in the case of the SQL Temp DB table, is the table instance name.</td>
</tr>
<tr>
<td>getPresenceFieldData</td>
<td>Retrieves the PresenceInfo value from the specified field.</td>
</tr>
<tr>
<td>getSQLStatement</td>
<td>Gets the SQL statement that is used to return records from the database.</td>
</tr>
<tr>
<td>getTableInInstanceHierarchy</td>
<td></td>
</tr>
<tr>
<td>getType</td>
<td>Indicates the type of the table.</td>
</tr>
<tr>
<td>helpField</td>
<td>Retrieves a string that contains the Help text for the specified field.</td>
</tr>
<tr>
<td>initValue</td>
<td>Initializes a field to the default value.</td>
</tr>
<tr>
<td>inputStatus</td>
<td>Sets or returns the current input status of the table buffer.</td>
</tr>
<tr>
<td>insert</td>
<td>Inserts the record into the table.</td>
</tr>
<tr>
<td>interactiveContext</td>
<td>Sets or returns the current interactive context of the table buffer.</td>
</tr>
<tr>
<td>isFieldDataRetrieved</td>
<td>Checks whether the data of the given field has been retrieved.</td>
</tr>
<tr>
<td>isFieldSet</td>
<td>Checks whether a field has a Set or Defaulted state.</td>
</tr>
<tr>
<td>isFormDataSource</td>
<td>Indicates whether the data source is a form.</td>
</tr>
<tr>
<td>isNewRecord</td>
<td>Returns true if the record is a new record that hasn’t been persisted yet.</td>
</tr>
<tr>
<td>isPartOfUOWSaveChanges</td>
<td></td>
</tr>
<tr>
<td>isTempDb</td>
<td>Indicates whether the type of the table is SQL TempDB.</td>
</tr>
<tr>
<td>isTmp</td>
<td>Indicates whether this is a temporary table.</td>
</tr>
<tr>
<td>joinChild</td>
<td>Finds the join child of the current record.</td>
</tr>
<tr>
<td>joinParent</td>
<td>Finds the join parent of the current record.</td>
</tr>
<tr>
<td>METHOD</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>linkPhysicalTableInstance</td>
<td>Checks whether there is a link for the physical table instance for the record.</td>
</tr>
<tr>
<td>merge</td>
<td>Merges the current table with the specified table.</td>
</tr>
<tr>
<td>modifiedField</td>
<td>Modifies the specified field to the original.</td>
</tr>
<tr>
<td>modifiedFieldValue</td>
<td>Modifies the specified field to the original value.</td>
</tr>
<tr>
<td>orig</td>
<td>Retrieves the original values of the current record.</td>
</tr>
<tr>
<td>overwriteSystemFields</td>
<td>Gets and sets the property that indicates whether system fields can be overwritten.</td>
</tr>
<tr>
<td>postLoad</td>
<td>Is executed after a record is read.</td>
</tr>
<tr>
<td>queryTimedOut</td>
<td>Indicates whether the query exceeded the time limit for execution.</td>
</tr>
<tr>
<td>queryTimeout</td>
<td>Gets and sets the property that indicates the time limit for the execution of a query.</td>
</tr>
<tr>
<td>readCommittedLock</td>
<td></td>
</tr>
<tr>
<td>readPast</td>
<td>Gets and sets the property that indicates whether to skip rows that are locked by other processes when a record is read.</td>
</tr>
<tr>
<td>recordLevelSecurity</td>
<td>Gets and sets the property that indicates whether to apply security on a record level.</td>
</tr>
<tr>
<td>relatedTable</td>
<td>Sets or returns the related buffer of a link of a table buffer.</td>
</tr>
<tr>
<td>hasRelatedTable</td>
<td>Indicates whether a foreign key constraint buffer is linked with the table.</td>
</tr>
<tr>
<td>renamePrimaryKey</td>
<td>Renames the foreign keys in other tables according to the change of the corresponding primary key value in this table.</td>
</tr>
<tr>
<td>reread</td>
<td>Rereads the record from the table.</td>
</tr>
<tr>
<td>RowCount</td>
<td>Retrieves the number of rows in the table.</td>
</tr>
<tr>
<td>selectForUpdate</td>
<td>Gets and sets the property that indicates whether to select records for update when they are read.</td>
</tr>
<tr>
<td>selectLocked</td>
<td>Indicates whether to select locked records.</td>
</tr>
<tr>
<td>selectRefRecord</td>
<td>Selects the record by referenced field ID.</td>
</tr>
<tr>
<td>selectWithRepeatableRead</td>
<td>Gets and sets the property that indicates whether repeatable read is enabled.</td>
</tr>
<tr>
<td>METHOD</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>setConnection</td>
<td>Sets the user connection for this table.</td>
</tr>
<tr>
<td>setCrossPartition</td>
<td>Sets or resets cross-partitioning for the table.</td>
</tr>
<tr>
<td>setFieldValue</td>
<td>Sets the field value in the record buffer.</td>
</tr>
<tr>
<td>setSQLTracing</td>
<td>Enables or disables SQL tracing mode.</td>
</tr>
<tr>
<td>setTempDB</td>
<td></td>
</tr>
<tr>
<td>setTmp</td>
<td>Sets the table so that it is not persisted to the database.</td>
</tr>
<tr>
<td>setTmpData</td>
<td>Sets the contents of the temporary table to the specified data.</td>
</tr>
<tr>
<td>setXDSContext</td>
<td>Sets new XDS context.</td>
</tr>
<tr>
<td>skipDatabaseLog</td>
<td>Gets and sets the property that indicates whether to skip database log requests.</td>
</tr>
<tr>
<td>skipDataMethods</td>
<td>Gets and sets the property that indicates whether to discard overloaded methods.</td>
</tr>
<tr>
<td>skipDeleteActions</td>
<td>Gets and sets the property that indicates whether to skip delete actions on the table.</td>
</tr>
<tr>
<td>skipDeleteMethod</td>
<td>Gets and sets the property that indicates whether to discard overloaded methods.</td>
</tr>
<tr>
<td>skipEvents</td>
<td>Provides an option to turn off calling the Application.event* methods for the lifetime of an xRecord object.</td>
</tr>
<tr>
<td>skipPostLoad</td>
<td>Gets and sets the property that indicates whether to skip executing the xRecord.postLoad method on the table.</td>
</tr>
<tr>
<td>skipTTSCheck</td>
<td>Gets and sets the property that indicates whether to skip the check to determine whether the record is selected for update.</td>
</tr>
<tr>
<td>suppressWarnings</td>
<td>Gets and sets the property that indicates whether to suppress warnings for this pointer.</td>
</tr>
<tr>
<td>tableAccessRight</td>
<td>Returns the table access right.</td>
</tr>
<tr>
<td>tableBufferAccessRight</td>
<td>Returns the table access right for the current record.</td>
</tr>
<tr>
<td>toolTipField</td>
<td>Retrieves the HelpText value for the specified field.</td>
</tr>
<tr>
<td>toolTipRecord</td>
<td>Retrieves the ToolTip value for the current record.</td>
</tr>
<tr>
<td>ttsabort</td>
<td>Aborts a transaction that was started by a call to the ttsbegin method.</td>
</tr>
<tr>
<td>METHOD</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ttsbegin</td>
<td>Starts a transaction that can be either committed by the ttscommit method or aborted by the ttsabort method.</td>
</tr>
<tr>
<td>ttscommit</td>
<td>Commits a transaction that was started by a call to the ttsbegin method.</td>
</tr>
<tr>
<td>update</td>
<td>Updates the current record.</td>
</tr>
<tr>
<td>validateDelete</td>
<td>Determines whether the current record is valid and ready to be deleted from the database.</td>
</tr>
<tr>
<td>validateField</td>
<td>Determines whether the specified field is valid.</td>
</tr>
<tr>
<td>validateFieldValue</td>
<td></td>
</tr>
<tr>
<td>validateRelations</td>
<td></td>
</tr>
<tr>
<td>validateWrite</td>
<td>Determines whether the current record is valid and ready to be written.</td>
</tr>
<tr>
<td>validTimeStateUpdateMode</td>
<td>Sets a valid time state update mode on the cursor.</td>
</tr>
<tr>
<td>wasCached</td>
<td>Specifies the location from which the data was retrieved.</td>
</tr>
<tr>
<td>write</td>
<td>Updates a record if it exists; otherwise, inserts a record.</td>
</tr>
<tr>
<td>xml</td>
<td>Retrieves an XML string that represents the current object.</td>
</tr>
<tr>
<td>takeOwnershipOfTempDBTable</td>
<td></td>
</tr>
<tr>
<td>useExistingTempDBTable</td>
<td></td>
</tr>
</tbody>
</table>

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreatedBy</td>
<td>String</td>
<td>CreatedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CreatedDateTime</td>
<td>UtcDateTime</td>
<td>CreatedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CreatedTransactionId</td>
<td>Int64</td>
<td>CreatedTransactionId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dataAreaId</td>
<td>String</td>
<td>DataAreaId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEL_CreatedTime</td>
<td>Int</td>
<td>DEL_CreatedTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEL_ModifiedTime</td>
<td>Int</td>
<td>DEL_ModifiedTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ModifiedBy</td>
<td>String</td>
<td>ModifiedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ModifiedDateTime</td>
<td>UtcDateTime</td>
<td>ModifiedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------</td>
<td>---------------</td>
<td>------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>ModifiedTransactionId</td>
<td>Int64</td>
<td>ModifiedTransactionId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Partition</td>
<td>Int64</td>
<td>Partition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReclId</td>
<td>Int64</td>
<td>ReclId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RelationType</td>
<td>Int64</td>
<td>RelationType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RowNumber</td>
<td>Int</td>
<td>RowNumber</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SequenceNum</td>
<td>Int64</td>
<td>SequenceNum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TableId</td>
<td>Int</td>
<td>TableId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UnionAllBranchId</td>
<td>Int</td>
<td>UnionAllBranchId</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataAreald</td>
<td>DataArea</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReclId</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Inheritance Hierarchy**

*xRecord Class Common Table*

**DataArea**

The DataArea table contains a list of companies that have been created in the database.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>alwaysNative</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>String</td>
<td>DataAreald</td>
<td></td>
<td>ID for an area of data</td>
</tr>
<tr>
<td>isVirtual</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>UserIdStr</td>
<td></td>
<td>Name</td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>---------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Partition</td>
<td>Int64</td>
<td>Partition</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>timeZone</td>
<td>Enum</td>
<td>Timezone</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>DataArea</td>
</tr>
<tr>
<td>Partition</td>
<td>Partitions</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>IdOnly</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table DataArea Table

**DatabaseLog**

The DatabaseLog table stores configuration information for the SysDatabaseLog table.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>createdBy</td>
<td>String</td>
<td>CreatedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td>CreatedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------</td>
<td>---------------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>dEL_CreatedTime</td>
<td>Integer</td>
<td>DEL_CreatedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>dEL_ModifiedTime</td>
<td>Integer</td>
<td>DEL_ModifiedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>domainId</td>
<td>String</td>
<td>DomainId</td>
<td></td>
<td>ID for the domain</td>
</tr>
<tr>
<td>logField</td>
<td>Integer</td>
<td>FieldId</td>
<td></td>
<td>ID for the field</td>
</tr>
<tr>
<td>logTable</td>
<td>Integer</td>
<td>TableId</td>
<td></td>
<td>ID for the table</td>
</tr>
<tr>
<td>logType</td>
<td>Enum</td>
<td></td>
<td>DatabaseLogType</td>
<td></td>
</tr>
<tr>
<td>modifiedBy</td>
<td>String</td>
<td>ModifiedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>modifiedDateTime</td>
<td>UtcDateTime</td>
<td>ModifiedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Field Groups

Field Group

logFieldRelation

Relations

Relation

Relation_DatabaseLog

DEL_DomainInfo

Indexes

Index

ALLOW DUPLICATES

FIELDS

Loglist

No

RecId

No

Security Note
Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

- **xRecord Class**
- **Common Table**
- **DatabaseLog Table**

**DEL_AccessRightsList**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>accessType</td>
<td>Enum</td>
<td></td>
<td>AccessType</td>
<td></td>
</tr>
<tr>
<td>accessTypeFkeyUse</td>
<td>Enum</td>
<td></td>
<td>AccessType</td>
<td></td>
</tr>
<tr>
<td>createdBy</td>
<td>String</td>
<td>CreatedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td>CreatedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dEL_CreatedTime</td>
<td>Integer</td>
<td>DEL_CreatedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>dEL_ModifiedTime</td>
<td>Integer</td>
<td>DEL_ModifiedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>domainId</td>
<td>String</td>
<td>DomainId</td>
<td></td>
<td>ID for the domain</td>
</tr>
<tr>
<td>elementName</td>
<td>String</td>
<td>UtilElementName</td>
<td></td>
<td>Name of the application element.</td>
</tr>
<tr>
<td>groupId</td>
<td>String</td>
<td>UserGroupId</td>
<td></td>
<td>ID for the user group</td>
</tr>
<tr>
<td>id</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>modifiedBy</td>
<td>String</td>
<td>ModifiedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------</td>
<td>----------------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>modifiedDateTime</td>
<td>UtcDateTime</td>
<td>ModifiedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>parentId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recl</td>
<td>Int64</td>
<td>Recl</td>
<td>AccessRecordType</td>
<td></td>
</tr>
<tr>
<td>recordType</td>
<td>Enum</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation_AccessRightsList1</td>
<td>UtilIdElements</td>
</tr>
<tr>
<td>Relation_AccessRightsList2</td>
<td>UtilIdElements</td>
</tr>
<tr>
<td>Relation_AccessRightsList3</td>
<td>DEL_DomainInfo</td>
</tr>
<tr>
<td>Relation_AccessRightsList4</td>
<td>DEL_UserGroupInfo</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Group</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Recl</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table DEL-AccessRightsList Table

**DEL_CompanyDomainList**

The CompanyDomainList table contains associations between the DomainInfo and DataArea tables. Security rights are granted per domain.

**Fields**
<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>companyId</td>
<td>String</td>
<td>SelectableDataArea</td>
<td></td>
<td>ID for the company you can select</td>
</tr>
<tr>
<td>createdBy</td>
<td>String</td>
<td>CreatedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td>CreatedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dEL_CreatedTime</td>
<td>Integer</td>
<td>DEL_CreatedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>dEL_ModifiedTime</td>
<td>Integer</td>
<td>DEL_ModifiedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>domainId</td>
<td>String</td>
<td>DomainId</td>
<td></td>
<td>ID for the domain</td>
</tr>
<tr>
<td>modifiedBy</td>
<td>String</td>
<td>ModifiedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>modifiedDateTime</td>
<td>UtcDateTime</td>
<td>ModifiedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>companyId</td>
<td>DataArea</td>
</tr>
<tr>
<td>domainId</td>
<td>DEL_DomainInfo</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Domain</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the
AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

**xRecord Class Common Table** DEL_CompanyDomainList Table

**DEL_DomainInfo**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>DomainId</td>
<td></td>
<td>ID for the domain</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>UserIdStr</td>
<td></td>
<td>Name</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>DEL_DomainInfo</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

**xRecord Class Common Table** DEL_DomainInfo Table

**DEL_UserGroupInfo**

The UserGroupInfo table contains the list of available user groups.

**Fields**
### Relations

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>UserGroupInfo</td>
</tr>
</tbody>
</table>

### Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

### Inheritance Hierarchy

**xRecord Class Common Table** DEL_UserGroupInfo Table

### DEL_UserGroupList

The UserGroupList table contains the list of users associated with each user groups.

### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>createdBy</td>
<td>String</td>
<td>CreatedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td>CreatedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dEL_CreatedTime</td>
<td>Integer</td>
<td>DEL_CreatedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>dEL_ModifiedTime</td>
<td>Integer</td>
<td>DEL_ModifiedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>groupId</td>
<td>String</td>
<td>UserGroupId</td>
<td></td>
<td>ID for the user group</td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>---------------</td>
<td>------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>modifiedBy</td>
<td>String</td>
<td></td>
<td>ModifiedBy</td>
<td></td>
</tr>
<tr>
<td>modifiedDateTime</td>
<td>UtcDateTime</td>
<td></td>
<td>ModifiedDateTime</td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td></td>
<td>RecId</td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td>RecVersion</td>
<td></td>
</tr>
<tr>
<td>userId</td>
<td>String</td>
<td></td>
<td>UserId</td>
<td>ID for the user</td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation_UserGroupList1</td>
<td>DEL_UserGroupInfo</td>
</tr>
<tr>
<td>Relation_UserGroupList2</td>
<td>UserInfo</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GroupId</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>UserId</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Inheritance Hierarchy**

xRecord Class Common Table DEL_UserGroupList Table

**ModelSecPolRuntimeEx**

The ModelSecPolRuntimeEx table stores the runtime metadata that is necessary to apply security policies.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConstrainedTable</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ContextString</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ContextType</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEL_ElementHandle</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEL_IsEnabled</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEL_LayerId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------</td>
<td>---------------</td>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>ElementHandle</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsDirty</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsEnabled</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsModeled</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LayerId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ModeledQueryDebugInfo</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ModeledQueryPackData</td>
<td>Container</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operation</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PrimaryTableAOTName</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QueryObjectAOTName</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConstrainedTableIdx</td>
<td>Yes</td>
<td>ConstrainedTable</td>
</tr>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td>RecId</td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table ModelSecPolRuntimeEx Table

**ModelSecPolRuntimeView**

The ModelSecPolRuntimeView view shows the runtime metadata for the currently active security policies.
### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConstrainedTable</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ContextString</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ContextType</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ElementHandle</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsDirty</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsModeled</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LayerId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ModeledQueryDebugInfo</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ModeledQueryPackData</td>
<td>Container</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operation</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PrimaryTableAOTName</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QueryObjectAOTName</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td></td>
<td>RecId</td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td>RecVersion</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table ModelSecPolRuntimeView Table

**Partitions**

The Partitions table contains the list of data partitions in the system.

**Fields**
<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>createdBy</td>
<td>String</td>
<td></td>
<td>CreatedBy</td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td></td>
<td>CreatedDateTime</td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>dEL_CreatedTime</td>
<td>Integer</td>
<td></td>
<td>DEL_CreatedTime</td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>dEL_ModifiedTime</td>
<td>Integer</td>
<td></td>
<td>DEL_ModifiedTime</td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>modifiedBy</td>
<td>String</td>
<td></td>
<td>ModifiedBy</td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>modifiedDateTime</td>
<td>UtcDateTime</td>
<td></td>
<td>ModifiedDateTime</td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td></td>
<td>UserIdStr</td>
<td>Name (This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------</td>
<td>-------</td>
<td>---------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PartitionKey</td>
<td>String</td>
<td>PartitionKey</td>
<td></td>
<td>Partition Key (This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>Recl</td>
<td>Int64</td>
<td>Recl</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
</tbody>
</table>

**Field Groups**

**FIELD GROUP**

<table>
<thead>
<tr>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autoidentification</td>
</tr>
</tbody>
</table>

**Relations**

**RELATION**

<table>
<thead>
<tr>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitions</td>
</tr>
</tbody>
</table>

**Indexes**

**INDEX**

<table>
<thead>
<tr>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table Partitions Table
# PrintJobHeader

The PrintJobHeader table contains information regarding the current print job.

## Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Extended Type</th>
<th>Enumeration Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>createdBy</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td></td>
<td>CreatedDateTime</td>
<td></td>
</tr>
<tr>
<td>dataAreaId</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dEL_CreatedTime</td>
<td>Integer</td>
<td>Integer</td>
<td>DEL_CreatedTime</td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>deviceName</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>format</td>
<td>Enum</td>
<td></td>
<td>PrintFormat</td>
<td></td>
</tr>
<tr>
<td>jobDescription</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>jobStatus</td>
<td>Enum</td>
<td></td>
<td>PrintJobStatus</td>
<td></td>
</tr>
<tr>
<td>jobType</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>numberOfPages</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Partition</td>
<td>Int64</td>
<td>Int64</td>
<td>Partition</td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>printedBy</td>
<td>String</td>
<td>UserId</td>
<td></td>
<td>ID for the user</td>
</tr>
<tr>
<td>printedDate</td>
<td>date</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>printedTime</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>printerInfo</td>
<td>Container</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>printFromPage</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>printNumcopies</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>printOnServer</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>DESCRIPTION</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------</td>
<td>---------------</td>
<td>-------------------</td>
<td></td>
</tr>
<tr>
<td>printToPage</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>unlimitedPageHeight</td>
<td>Enum</td>
<td>boolean</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataAreald</td>
<td>DataArea</td>
</tr>
<tr>
<td>Partition</td>
<td>Partitions</td>
</tr>
<tr>
<td>printedBy</td>
<td>UserInfo</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreatedBy</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>CreatedDate</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>JobType</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Inheritance Hierarchy**

xRecord Class Common Table PrintJobHeader Table

**PrintJobPages**

The PrintJobPages table contains information regarding the currently printing page of a print job

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataAreald</td>
<td>String</td>
<td>DataAreald</td>
<td></td>
</tr>
<tr>
<td>numberOfLines</td>
<td>Int</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pageContents</td>
<td>Container</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pageNo</td>
<td>Int</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pagesHeaderRecId</td>
<td>Int64</td>
<td>RecId</td>
<td>Unique ID for the record in the database</td>
</tr>
</tbody>
</table>
SecurableObject

The SecurableObject table contains all security artifacts reference by the security framework.

Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChildName</td>
<td>String</td>
<td>SecurableChildName</td>
<td></td>
<td>The child name of the securable object.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>SecurableName</td>
<td></td>
<td>The name of the securable object.</td>
</tr>
</tbody>
</table>
Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Security Note**

**Inheritance Hierarchy**

xRecord Class Common Table SecurableObject Table
<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td>RecId</td>
</tr>
<tr>
<td>IdentifierIdx</td>
<td>No</td>
<td>Identifier</td>
</tr>
<tr>
<td>NameIdx</td>
<td>Yes</td>
<td>Name</td>
</tr>
</tbody>
</table>

**Inheritance Hierarchy**

xRecord Class Common Table SecurityDuty Table

**SecurityEntryPointInferredTables**

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntryPointName</td>
<td>String</td>
<td>SecurableName</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Enum</td>
<td></td>
<td>SecurableType</td>
<td></td>
</tr>
<tr>
<td>TableName</td>
<td>String</td>
<td>SecurableName</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AllowEdit</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>AllowCreate</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>AllowDelete</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>ValidTimeStateUpdate</td>
<td>Enum</td>
<td></td>
<td>ValidTimeStateUpdate</td>
<td></td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td>RecId</td>
</tr>
<tr>
<td>EntryPointTableIdx</td>
<td>No</td>
<td>EntryPointName, Type, TableName, ValidTimeStateUpdate</td>
</tr>
</tbody>
</table>

**Inheritance Hierarchy**

xRecord Class Common Table SecurityEntryPointInferredTables Table

**SecurityEntryPointLink**

The SecurityEntryPointLink table contains the entry point to securable object mapping that has been specified on the AOT nodes of menu items and web menu items.

**Fields**
<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntryPoint</td>
<td>Int64</td>
<td>Recld</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>PermissionOwner</td>
<td>Int64</td>
<td>Recld</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>Recld</td>
<td>Int64</td>
<td>Recld</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>ReclVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ValidFrom</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ValidTo</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation_SecurityEntryPointLink1</td>
<td>SecurableObject</td>
</tr>
<tr>
<td>Relation_SecurityEntryPointLink2</td>
<td>SecurableObject</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntryPointIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ReclDIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SecurityEntryPointLink Table

**SecurityPermission**

The SecurityPermission table contains the list of permissions that have been specified on the AOT nodes of forms, reports, security code permissions, and service operations.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Enum</td>
<td></td>
<td>AccessRight</td>
<td></td>
</tr>
</tbody>
</table>
### Group

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Owner</td>
<td>Int64</td>
<td>RecId</td>
<td>AccessRight</td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecurableObject</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>ValidFrom</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ValidTo</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Relations

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation_SecurityPermission1</td>
<td>SecurableObject</td>
</tr>
<tr>
<td>Relation_SecurityPermission2</td>
<td>SecurableObject</td>
</tr>
</tbody>
</table>

### Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>OwnerGroupObjectIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

### Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

### Inheritance Hierarchy

xRecord Class Common Table SecurityPermission Table

### SecurityPrivilege

#### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>String</td>
<td>SecurityPrivilegeIdent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------</td>
<td>-----------------------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>SecurityPrivilegeName</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>String</td>
<td>SecurityPrivilegeDesc</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Field Groups**

<table>
<thead>
<tr>
<th>FIELD GROUP</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autoidentification</td>
<td>Name</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td>RecId</td>
</tr>
<tr>
<td>IdentifierIdx</td>
<td>No</td>
<td>Identifier</td>
</tr>
<tr>
<td>NameIdx</td>
<td>Yes</td>
<td>Name</td>
</tr>
</tbody>
</table>

**Inheritance Hierarchy**

xRecord Class Common Table SecurityPrivilege Table

**SecurityRole**

The SecurityRole table reflects the list of roles defined by the security AOT role node.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowCurrentRecords</td>
<td>Enum</td>
<td></td>
<td>AccessRight</td>
<td></td>
</tr>
<tr>
<td>AllowFutureRecords</td>
<td>Enum</td>
<td></td>
<td>AccessRight</td>
<td></td>
</tr>
<tr>
<td>AllowPastRecords</td>
<td>Enum</td>
<td></td>
<td>AccessRight</td>
<td></td>
</tr>
<tr>
<td>AotName</td>
<td>String</td>
<td>SecurityRoleAotName</td>
<td>AccessRight</td>
<td>The name of the role in the AOT.</td>
</tr>
<tr>
<td>ContextString</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEL_AllowCurrentRecords</td>
<td>Enum</td>
<td></td>
<td>AccessRight</td>
<td></td>
</tr>
<tr>
<td>DEL_AllowFutureRecords</td>
<td>Enum</td>
<td></td>
<td>AccessRight</td>
<td></td>
</tr>
<tr>
<td>DEL_AllowPastRecords</td>
<td>Enum</td>
<td></td>
<td>AccessRight</td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------</td>
<td>-------------------</td>
<td>------------------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>DEL_IsEnabled</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td>Description of the security role.</td>
</tr>
<tr>
<td>Description</td>
<td>String</td>
<td>SecurityRoleDescription</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsEnabled</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>SecurityRoleName</td>
<td></td>
<td>The name of the security role.</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UserLicenseType</td>
<td>Enum</td>
<td></td>
<td>UserLicenseType</td>
<td></td>
</tr>
</tbody>
</table>

**Field Groups**

<table>
<thead>
<tr>
<th>FIELD GROUP</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autoidentification</td>
<td></td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AotNameIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>NameIdx</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>RecIDIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SecurityRole Table

**SecurityRoleAssignmentRule**

Rules for dynamically assigning users to role

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------</td>
<td>---------------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MembershipRuleDescription</td>
<td>String</td>
<td>MembershipRuleDescription</td>
<td></td>
<td>Description of the automatic role membership rule</td>
</tr>
<tr>
<td>MembershipRuleName</td>
<td>String</td>
<td>MembershipRuleName</td>
<td></td>
<td>Name of the automatic role membership rule</td>
</tr>
<tr>
<td>Partition</td>
<td>Int64</td>
<td>Partition</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
</tbody>
</table>

| RecId         | Int64     | RecId               |                |                                                                            |
| recVersion    | Integer   | RecVersion          |                |                                                                            |
| RuleQuery     | Container |                     |                |                                                                            |
| SecurityRole  | Int64     | RecId               |                | Unique ID for the record in the database                                   |
| ValidFrom     | UtcDateTime |                    |                |                                                                            |
| ValidTo       | UtcDateTime |                    |                |                                                                            |

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition</td>
<td>Partitions</td>
</tr>
<tr>
<td>SecurityRole</td>
<td>SecurityRole</td>
</tr>
<tr>
<td>SecurityRoleRelationship</td>
<td>SecurityRole</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlternateKey</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RecIDIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not
authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SecurityRoleAssignmentRule Table

---

### SecurityRoleDutyExplodedGraph

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecurityRole</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecurityDuty</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecurityRole</td>
<td>SecurityRole</td>
</tr>
<tr>
<td>SecurityDuty</td>
<td>SecurityDuty</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td>RecId</td>
</tr>
<tr>
<td>RoleDutyIdx</td>
<td>No</td>
<td>SecurityRole, SecurityDuty</td>
</tr>
</tbody>
</table>

---

**Inheritance Hierarchy**

xRecord Class Common Table SecurityRoleDutyExplodedGraph Table

---

### SecurityRoleExplodedGraph

The SecurityRoleExplodedGraph table contains all role relationships, direct or indirect, as defined by the AOT sub role nodes of the security role nodes.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RefCount</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecurityRole</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------</td>
<td>---------------</td>
<td>------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>SecuritySubRole</td>
<td>Int64</td>
<td>Recld</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation_SecurityRole1</td>
<td>SecurityRole</td>
</tr>
<tr>
<td>Relation_SecurityRole2</td>
<td>SecurityRole</td>
</tr>
<tr>
<td>SecurityRole</td>
<td>SecurityRole</td>
</tr>
<tr>
<td>SecuritySubRole</td>
<td>SecurityRole</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RoleSubRoleIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SecurityRoleExplodedGraph Table

**SecurityRolePermissionOverride**

The SecurityRolePermissionOverride table contains the list of permissions that have been specified on the security role AOT nodes.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Enum</td>
<td></td>
<td>AccessRight</td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>Recld</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecurableObject</td>
<td>Int64</td>
<td>Recld</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------</td>
<td>---------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SecurityRole</td>
<td>Int64</td>
<td></td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>ValidFrom</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ValidTo</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation_SecurityRolePermissionOverride1</td>
<td>SecurityRole</td>
</tr>
<tr>
<td>Relation_SecurityRolePermissionOverride2</td>
<td>SecurableObject</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RoleObjectIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SecurityRolePermissionOverride Table

**SecurityRolePrivilegeExplodedGraph**

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecurityRole</td>
<td>Int64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecurityPrivilege</td>
<td>Int64</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecurityRole</td>
<td>SecurityRole</td>
</tr>
<tr>
<td>SecurityPrivilege</td>
<td>SecurityPrivilege</td>
</tr>
</tbody>
</table>
## Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td>RecId</td>
</tr>
<tr>
<td>RolePrivilegeIdx</td>
<td>No</td>
<td>SecurityRole, SecurityPrivilege</td>
</tr>
</tbody>
</table>

## Inheritance Hierarchy

xRecord Class Common Table SecurityRolePrivilegeExplodedGraph Table

## SecurityRoleRuntime

### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecurityRole</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>SecurableName</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ChildName</td>
<td>String</td>
<td>SecurableChildName</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Enum</td>
<td></td>
<td>SecurableType</td>
<td></td>
</tr>
<tr>
<td>CreateAccess</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReadAccess</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UpdateAccess</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeleteAccess</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CorrectAccess</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>InvokeAccess</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PastCreateAccess</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PastReadAccess</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PastUpdateAccess</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PastDeleteAccess</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PastCorrectAccess</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PastInvokeAccess</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CurrentCreateAccess</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CurrentReadAccess</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CurrentUpdateAccess | Int |  |  |  | 
CurrentDeleteAccess | Int |  |  |  | 
CurrentCorrectAccess | Int |  |  |  | 
CurrentInvoke | Int |  |  |  | 
FutureCreateAccess | Int |  |  |  | 
FutureReadAccess | Int |  |  |  | 
FutureUpdateAccess | Int |  |  |  | 
FutureDeleteAccess | Int |  |  |  | 
FutureCorrectAccess | Int |  |  |  | 
FutureInvokeAccess | Int |  |  |  | 

**Field Groups**

**FIELD GROUP** | **FIELDS**
--- | ---
AutoIdentification | Name, ChildName, Type

**Indexes**

**INDEX** | **ALLOW DUPLICATES** | **FIELDS**
--- | --- | ---
RecIdIdx | No | RecId
RoleIDIdx | Yes | SecurityRole
SecureableObjectIdx | Yes | Type, Name, ChildName

**Inheritance Hierarchy**

xRecord Class Common Table SecurityRoleRuntime Table

**SecurityRoleTaskGrant**

The SecurityRoleTaskGrant table contains the list of role to duty mappings and role to privilege mappings as defined by the AOT security role node.

**Fields**

**FIELD** | **TYPE** | **EXTENDED TYPE** | **ENUMERATION TYPE** | **DESCRIPTION**
--- | --- | --- | --- | ---
RecId | Int64 | RecId |  |  |
recVersion | Integer | RecVersion |  |  |
SecurityRole

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecurityRole</td>
<td>Int64</td>
<td>Recld</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
</tbody>
</table>

SecurityTask

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SecurityTask</td>
<td>Int64</td>
<td>Recld</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
</tbody>
</table>

Relations

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation_SecurityRoleTaskGrant1</td>
<td>SecurityRole</td>
</tr>
<tr>
<td>Relation_SecurityRoleTaskGrant2</td>
<td>SecurityTask</td>
</tr>
</tbody>
</table>

Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RoleTaskIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SecurityRoleTaskGrant Table

SecuritySegregationOfDutiesConflict

The SecuritySegregationOfDutiesConflict table stores information about segregation of duties conflicts that result from attempted assignments of users to roles, and resolutions to the conflicts provided by authorized users.

Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssignmentMode</td>
<td>Enum</td>
<td></td>
<td>RoleAssignmentMode</td>
<td></td>
</tr>
<tr>
<td>createdBy</td>
<td>String</td>
<td>CreatedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td>CreatedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>------------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>dEL_CreatedTime</td>
<td>Integer</td>
<td>DEL_CreatedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>DEL_ExistingTask</td>
<td>Int64</td>
<td>ReclId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dEL_ModifiedTime</td>
<td>Integer</td>
<td>DEL_ModifiedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>DEL_NewTask</td>
<td>Int64</td>
<td>ReclId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ExistingDuty</td>
<td>Int64</td>
<td>ReclId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ExistingRole</td>
<td>Int64</td>
<td>ReclId</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>ExistingTask</td>
<td>Int64</td>
<td>ReclId</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>modifiedBy</td>
<td>String</td>
<td>ModifiedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>modifiedDateTime</td>
<td>UtcDateTime</td>
<td>ModifiedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NewDuty</td>
<td>Int64</td>
<td>ReclId</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>NewRole</td>
<td>Int64</td>
<td>ReclId</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>NewTask</td>
<td>Int64</td>
<td>ReclId</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>Partition</td>
<td>Int64</td>
<td>Partition</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------</td>
<td>--------------------------------</td>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ReasonForOverride</td>
<td>VarString</td>
<td>SegregationOfDuties OverrideComment</td>
<td></td>
<td>Comment explaining the reason for overriding the segregation of duties violation</td>
</tr>
<tr>
<td>Recl</td>
<td>Int64</td>
<td>Recl</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resolution</td>
<td>Enum</td>
<td></td>
<td>SegregationOfDuties Resolution</td>
<td></td>
</tr>
<tr>
<td>SegregationOfDuties Rule</td>
<td>Int64</td>
<td>Recl</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>User</td>
<td>String</td>
<td>UserId</td>
<td></td>
<td>ID for the user</td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>Relation</th>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExistingDuty</td>
<td>SecurityDuty</td>
</tr>
<tr>
<td>ExistingRole</td>
<td>SecurityRole</td>
</tr>
<tr>
<td>ExistingRoleRelationship</td>
<td>SecurityRole</td>
</tr>
<tr>
<td>ExistingTaskRelationship</td>
<td>SecurityTask</td>
</tr>
<tr>
<td>NewDuty</td>
<td>SecurityDuty</td>
</tr>
<tr>
<td>NewRole</td>
<td>SecurityRole</td>
</tr>
<tr>
<td>NewRoleRelationship</td>
<td>SecurityRole</td>
</tr>
<tr>
<td>NewTaskRelationship</td>
<td>SecurityTask</td>
</tr>
<tr>
<td>Partition</td>
<td>Partitions</td>
</tr>
<tr>
<td>Relation_SecuritySegregation7</td>
<td>UserInfo</td>
</tr>
<tr>
<td>SecuritySODRuleRelationship</td>
<td>SecuritySegregationOfDutiesRule</td>
</tr>
<tr>
<td>SegregationOfDutiesRule</td>
<td>SecuritySegregationOfDutiesRule</td>
</tr>
<tr>
<td>User</td>
<td>UserInfo</td>
</tr>
</tbody>
</table>

**Indexes**
Security Note
Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy
xRecord Class Common Table SecuritySegregationOfDutiesConflict Table

SecuritySegregationOfDutiesRule
The SecuritySegregationOfDutiesRule table stores the rules governing segregation of duties.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEL_FirstSecurityTask</td>
<td>Int64</td>
<td></td>
<td>RecId</td>
<td></td>
</tr>
<tr>
<td>DEL_SecondSecurityTask</td>
<td>Int64</td>
<td></td>
<td>RecId</td>
<td></td>
</tr>
<tr>
<td>FirstDuty</td>
<td>Int64</td>
<td></td>
<td>RecId</td>
<td></td>
</tr>
<tr>
<td>FirstSecurityTask</td>
<td>Int64</td>
<td></td>
<td>RecId</td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>Mitigation</td>
<td>String</td>
<td>SecurityMitigation</td>
<td></td>
<td>Mitigation for the risk associated with violating the segregation of duties rule</td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------</td>
<td>------------------------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>SegregationOfDuties</td>
<td>RuleName</td>
<td>Name of the segregation of duties rule</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Risk</td>
<td>String</td>
<td>SecurityRisk</td>
<td></td>
<td>Risk associated with violating the segregation of duties rule</td>
</tr>
<tr>
<td>SecondDuty</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecondSecurityTask</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>Severity</td>
<td>Enum</td>
<td></td>
<td>SegregationOfDuties</td>
<td>Severity</td>
</tr>
<tr>
<td>ValidFrom</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ValidTo</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Field Groups**

**FIELD GROUP**

<table>
<thead>
<tr>
<th>AutoIdentification</th>
</tr>
</thead>
</table>

**Relations**

**RELATION**

<table>
<thead>
<tr>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FirstDuty</td>
</tr>
<tr>
<td>FirstSecurityTaskRelationship</td>
</tr>
<tr>
<td>SecondDuty</td>
</tr>
<tr>
<td>SecondSecurityTaskRelationship</td>
</tr>
</tbody>
</table>

**Indexes**

**INDEX**

<table>
<thead>
<tr>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>FirstDuty</td>
</tr>
<tr>
<td>INDEX</td>
<td>ALLOW DUPLICATES</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------</td>
</tr>
<tr>
<td>NameIdx</td>
<td>No</td>
</tr>
<tr>
<td>RecIdIdx</td>
<td>No</td>
</tr>
<tr>
<td>SecondSecurityDuty</td>
<td>Yes</td>
</tr>
<tr>
<td>SecondSecurityTask</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SecuritySegregationOfDutiesRule Table

**SecuritySubRole**

The SecuritySubRole table contains all sub roles that have been specified on the security role AOT nodes.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecurityRole</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>SecuritySubRole</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>ValidFrom</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ValidTo</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation_SecuritySubRole1</td>
<td>SecurityRole</td>
</tr>
<tr>
<td>Relation_SecurityTaskPermission2</td>
<td>SecurityRole</td>
</tr>
<tr>
<td>SecurityRole</td>
<td>SecurityRole</td>
</tr>
<tr>
<td>Relation</td>
<td>Table</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>SecuritySubRole</td>
<td>SecurityRole</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>Index</th>
<th>Allow Duplicates</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RoleSubRoleIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SecuritySubRole Table

**SecuritySubTask**

The SecuritySubTask table contains the duty to privilege mappings that have been specified on the security duty AOT nodes.

**Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Extended Type</th>
<th>Enumeration Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecId</td>
<td>Int64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecuritySubTask</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecurityTask</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ValidFrom</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ValidTo</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>Relation</th>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation_SecuritySubTask1</td>
<td>SecurityTask</td>
</tr>
<tr>
<td>Relation_SecuritySubTask2</td>
<td>SecurityTask</td>
</tr>
</tbody>
</table>
### Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>TaskSubTaskIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

### Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SecuritySubTask Table

### SecurityTask

The SecurityTask table contains the list of duties and privileges that have been defined by the AOT security duty and security privilege nodes.

### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AotName</td>
<td>String</td>
<td>SecurityTaskAotName</td>
<td></td>
<td>The name of the task in the AOT.</td>
</tr>
<tr>
<td>Description</td>
<td>String</td>
<td>SecurityTaskDescrip</td>
<td></td>
<td>Description of the process cycle, duty, or privilege.</td>
</tr>
<tr>
<td>IsEnabled</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>IsPermissionSet</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>SecurityTaskName</td>
<td></td>
<td>The name of the process cycle, duty, or privilege.</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>Recld</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Enum</td>
<td>SecurityTaskType</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Field Groups

<table>
<thead>
<tr>
<th>FIELD GROUP</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autoidentification</td>
<td></td>
</tr>
</tbody>
</table>

### Indexes
### Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

### Inheritance Hierarchy

xARecord Class

Common Table

SecurityTask Table

### SecurityTaskEntryPoint

The SecurityTaskEntryPoint table contains the list of privilege to entry point mappings that have been specified on the AOT security privilege node.

### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>EntryPoint</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>PermissionGroup</td>
<td>Enum</td>
<td></td>
<td>AccessRight</td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecurityTask</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>ValidFrom</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ValidTo</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Relations

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation_SecurityTaskEntryPoint1</td>
<td>SecurityTask</td>
</tr>
<tr>
<td>Relation_SecurityTaskEntryPoint2</td>
<td>SecurableObject</td>
</tr>
</tbody>
</table>

### Indexes
### Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

### Inheritance Hierarchy

xRecord Class Common Table SecurityTaskEntryPoint Table

### SecurityTaskExplodedGraph

The SecurityTaskExplodedGraph table contains the duty to privilege mappings that have been specified on the security duty AOT nodes.

#### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RefCount</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecuritySubTask</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>SecurityTask</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
</tbody>
</table>

#### Relations

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation_SecurityTaskExplodedGraph1</td>
<td>SecurityTask</td>
</tr>
<tr>
<td>Relation_SecurityTaskExplodedGraph2</td>
<td>SecurityTask</td>
</tr>
</tbody>
</table>

#### Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>SubTaskIdx</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>INDEX</td>
<td>ALLOW DUPLICATES</td>
<td>FIELDS</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------</td>
<td>--------</td>
</tr>
<tr>
<td>TaskSubTaskIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SecurityTaskExplodedGraph Table

**SecurityTaskPermission**

The SecurityTaskPermission table is obsolete.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Level</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>Recld</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecurableObject</td>
<td>Int64</td>
<td>Recld</td>
<td>Unique ID for the record in the database</td>
<td></td>
</tr>
<tr>
<td>SecurityTask</td>
<td>Int64</td>
<td>Recld</td>
<td>Unique ID for the record in the database</td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation_SecurityTaskPermission1</td>
<td>SecurityTask</td>
</tr>
<tr>
<td>Relation_SecurityTaskPermission2</td>
<td>SecurableObject</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIDIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>TaskObjectIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
Security Note
Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the
AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object
Server authorizes each create, update, and delete action on the table by confirming that the current user has
permission to perform the requested operation on that table. If the user who initiates the operation is not
authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

Security Task Permission Override Table

SecurityTaskPermissionOverride
The SecurityTaskPermissionOverride table contains the list of permissions that have been specified on the
security privilege AOT nodes.

Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Enum</td>
<td></td>
<td>AccessRight</td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecurableObject</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>SecurityTask</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ValidFrom</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ValidTo</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Relations

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation_SecurityTaskPermissionOverride1</td>
<td>SecurityTask</td>
</tr>
<tr>
<td>Relation_SecurityTaskPermissionOverride2</td>
<td>SecurableObject</td>
</tr>
</tbody>
</table>

Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>TaskObjectIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Security Note
Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the
Inheritance Hierarchy

SecurityUserRole

The SecurityUserRole table contains the user to role mappings.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssignmentMode</td>
<td>Enum</td>
<td></td>
<td>RoleAssignmentMode</td>
<td>AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.</td>
</tr>
<tr>
<td>AssignmentStatus</td>
<td>Enum</td>
<td></td>
<td>RoleAssignmentStatus</td>
<td></td>
</tr>
<tr>
<td>Partition</td>
<td>Int64</td>
<td>Partition</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecurityRole</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>User</td>
<td>String</td>
<td>UserId</td>
<td></td>
<td>ID for the user</td>
</tr>
<tr>
<td>ValidFrom</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ValidTo</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Relations

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition</td>
<td>Partitions</td>
</tr>
<tr>
<td>Relation_SecurityRole</td>
<td>SecurityRole</td>
</tr>
<tr>
<td>Relation_SecurityUserRole3</td>
<td>UserInfo</td>
</tr>
<tr>
<td>SecurityRole</td>
<td>SecurityRole</td>
</tr>
</tbody>
</table>
**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>UserRoleIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

**SecurityUserRoleCondition**

The SecurityUserRoleCondition table contains the list of companies that constrain a user to role mappings. If there are no entries for a particular user to role mapping then the user is granted the permissions of that role for all companies.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ControllingKey</td>
<td>int64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DataArea</td>
<td>String</td>
<td>DataAreaId</td>
<td></td>
<td>ID for an area of data</td>
</tr>
<tr>
<td>Partition</td>
<td>Int64</td>
<td>Partition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecurityUserRole</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
</tbody>
</table>

**Relations**
### Relation

<table>
<thead>
<tr>
<th>Relation</th>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataArea</td>
<td>DataArea</td>
</tr>
<tr>
<td>Partition</td>
<td>Partitions</td>
</tr>
<tr>
<td>Relation_SecurityUserRoleCondition1</td>
<td>SecurityUserRole</td>
</tr>
<tr>
<td>Relation_SecurityUserRoleCondition2</td>
<td>DataArea</td>
</tr>
<tr>
<td>SecurityUserRole</td>
<td>SecurityUserRole</td>
</tr>
</tbody>
</table>

### Indexes

<table>
<thead>
<tr>
<th>Index</th>
<th>Allow Duplicates</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIDIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>UserRoleDataAreaIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

### Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

### Inheritance Hierarchy

xRecord Class Common Table SecurityUserRoleCondition Table

### SqlDescribe

The SqlDescribe table is used to store the table and field metadata. The SqlDataDictionary::tablemetadata method populates this table by using a back end database query.

### Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Extended Type</th>
<th>Enumeration Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fieldId</td>
<td>Integer</td>
<td>FieldId</td>
<td></td>
<td>ID for the field</td>
</tr>
<tr>
<td>fieldType</td>
<td>Enum</td>
<td></td>
<td>Types</td>
<td></td>
</tr>
<tr>
<td>flags</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>UtilElementName</td>
<td></td>
<td>Name of the application element.</td>
</tr>
<tr>
<td>nullable</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>numericPrecision</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The SqlDictionary table describes the current state of the database with respect to the table and field metadata. The table also contains view and table dependency information. The database synchronization engine uses the SqlDictionary table to determine the actions that are required to synchronize the AOT with the database.

### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fieldId</td>
<td>Integer</td>
<td>FieldId</td>
<td></td>
<td>ID for the field</td>
</tr>
<tr>
<td>fieldType</td>
<td>Enum</td>
<td></td>
<td></td>
<td>Types</td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>---------------</td>
<td>------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>flags</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>UtilElementName</td>
<td></td>
<td>Name of the application element.</td>
</tr>
<tr>
<td>nullable</td>
<td>Enum</td>
<td>boolean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rightJustify</td>
<td>Enum</td>
<td>boolean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>shadow</td>
<td>Enum</td>
<td>boolean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sqlName</td>
<td>String</td>
<td>UtilElementName</td>
<td></td>
<td>Name of the application element.</td>
</tr>
<tr>
<td>strSize</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tabId</td>
<td>Integer</td>
<td>TableId</td>
<td></td>
<td>ID for the table</td>
</tr>
</tbody>
</table>

Field Groups

<table>
<thead>
<tr>
<th>FIELD GROUP</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldIdRelation</td>
<td></td>
</tr>
</tbody>
</table>

Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SqlDictionary Table

SqlParameters

The SqlParameters table stores database related information in the form of parameter and value pairs. This table is not used in Microsoft Dynamics Ax 2009.

Fields
<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>iParm</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>iValue</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>parm</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td></td>
<td>RecId</td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td>RecVersion</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parm</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SqlParameters Table

**SqlStatistics**

The SqlStatistics table stores related database statistics for the user. This table is not used in Microsoft Dynamics Ax 2009.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>dEL_ModifiedTime</td>
<td>Integer</td>
<td>DEL_ModifiedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>indexId</td>
<td>Integer</td>
<td>IndexId</td>
<td></td>
<td>ID for the index</td>
</tr>
<tr>
<td>modifiedBy</td>
<td>String</td>
<td>ModifiedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>modifiedDateTime</td>
<td>UtcDateTime</td>
<td>ModifiedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
<td>---------------</td>
<td>------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>objectType</td>
<td>Enum</td>
<td></td>
<td>SqlStatType</td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tabId</td>
<td>Integer</td>
<td>TableId</td>
<td></td>
<td>ID for the table</td>
</tr>
<tr>
<td>userId</td>
<td>String</td>
<td>UserId</td>
<td></td>
<td>ID for the user</td>
</tr>
<tr>
<td>value1</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value10</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value11</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value12</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value2</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value3</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value4</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value5</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value6</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value7</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value8</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value9</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Field Groups**

<table>
<thead>
<tr>
<th>FIELD GROUP</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>indexIdRelation</td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation_SqlStats1</td>
<td>SqlStatistics</td>
</tr>
<tr>
<td>Relation_SqlStats2</td>
<td>UserInfo</td>
</tr>
<tr>
<td>tabId</td>
<td>SqlStatistics</td>
</tr>
<tr>
<td>userId</td>
<td>UserInfo</td>
</tr>
</tbody>
</table>
Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Inheritance Hierarchy

xRecord Class Common Table SqlStatistics Table

SqlStorage

The SqlStorage table contains information about table space and its Oracle attributes.

Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>indexId</td>
<td>Integer</td>
<td>IndexId</td>
<td></td>
<td>ID for the index</td>
</tr>
<tr>
<td>objectType</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>override</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>parm</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tabId</td>
<td>Integer</td>
<td>TableId</td>
<td></td>
<td>ID for the table</td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Field Groups

Field Group: indexIdRelation

Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Inheritance Hierarchy

xRecord Class Common Table SqlStorage Table

SqlSyncInfo
The SqlSyncInfo table captures messages and DDL statements during the database synchronization process. Once the synchronization process is complete the information in the table is deleted.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LogType</td>
<td>Enum</td>
<td></td>
<td>SqlSyncLogType</td>
<td></td>
</tr>
<tr>
<td>MessageType</td>
<td>Enum</td>
<td></td>
<td>SqlSyncMessageType</td>
<td></td>
</tr>
<tr>
<td>ParentID</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td></td>
<td>RecId</td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td>RecVersion</td>
<td></td>
</tr>
<tr>
<td>Sequence</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SyncTable</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>TableName</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WarningOk</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TableName</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateReadUpdateDelete. The Application Object Server authorizes each create, read, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SqlSyncInfo Table

**Subquery**

The Subquery table is used by position based paging functionality.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataAreaid</td>
<td>String</td>
<td>DataAreaid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>---------------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataAreaId</td>
<td>DataArea</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecId</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Inheritance Hierarchy**

*xRecord Class Common Table Subquery Table*

**SysActiveTempTable**

The SysActiveTempTable table provides data about the temporary database tables that are currently created. The table is used by the framework to manage the lifetime of these tables.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td></td>
<td>CreatedDateTime</td>
<td></td>
</tr>
<tr>
<td>dEL_CreatedTime</td>
<td>Integer</td>
<td></td>
<td>DEL_CreatedTime</td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>InstanceId</td>
<td>String</td>
<td></td>
<td>UtilElementName</td>
<td>Name of the application element.</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td></td>
<td>RecId</td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td>RecVersion</td>
<td></td>
</tr>
<tr>
<td>RelationTypeld</td>
<td>Int64</td>
<td></td>
<td>RecId</td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>ServerId</td>
<td>Int64</td>
<td></td>
<td>RecId</td>
<td>Unique ID for the record in the database</td>
</tr>
</tbody>
</table>
Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysActiveTempTable Table

SysBCProxyUserAccount

The SysBCProxyUserAccount table stores the business connector proxy information that is entered through the SysBcAliasForm security form. This table always contains one record.

Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>Enumeration Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>networkAlias</td>
<td>String</td>
<td>NetworkAlias</td>
<td></td>
<td></td>
</tr>
<tr>
<td>networkDomain</td>
<td>String</td>
<td>NetworkDomain</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>Recld</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sid</td>
<td>String</td>
<td>Sid</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecID</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Sid</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.
authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SysBCProxyUserAccount Table

### SysBreakpointList

The SysBreakpointList table contains a list of developers that have breakpoints in MorphX.

#### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>createdBy</td>
<td>String</td>
<td></td>
<td>CreatedBy</td>
<td></td>
</tr>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td></td>
<td>CreatedDateTime</td>
<td></td>
</tr>
<tr>
<td>dEL_CreatedTime</td>
<td>Integer</td>
<td></td>
<td>DEL_CreatedTime</td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>machineName</td>
<td>String</td>
<td></td>
<td>NetworkDomain</td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td></td>
<td>RecId</td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td>RecVersion</td>
<td></td>
</tr>
<tr>
<td>userId</td>
<td>String</td>
<td></td>
<td>UserId</td>
<td>ID for the user</td>
</tr>
<tr>
<td>version</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Relations

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>userId</td>
<td>UserInfo</td>
</tr>
</tbody>
</table>

#### Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecId</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>UserId</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

**Inheritance Hierarchy**

xRecord Class Common Table SysBreakpointList Table

### SysBreakpoints

The SysBreakpoints table contains a list of all the breakpoints in MorphX.
### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>codePath</td>
<td>Container</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lineNo</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>listRecId</td>
<td>Int64</td>
<td>Recl</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>Recl</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>status</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>version</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Relations

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>listRecId</td>
<td>SysBreakpointList</td>
</tr>
<tr>
<td>Relation_SysBreakpoints1</td>
<td>SysBreakpointList</td>
</tr>
</tbody>
</table>

### Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListRecId</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

### Inheritance Hierarchy

xRecord Class Common Table SysBreakpoints Table

### SysCacheFlush

The SysCacheFlush table contains data that is used for synchronization of caches across multiple AOS servers.

### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClearData</td>
<td>Container</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FlushData</td>
<td>Container</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FlushVersion</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>modifiedDateTime</td>
<td>UtcDateTime</td>
<td></td>
<td>ModifiedDateTime</td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>Recl</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scope</td>
<td>String</td>
<td>GlobalObjectCacheScope</td>
<td></td>
<td>Name of an instance in the global object cache.</td>
</tr>
</tbody>
</table>

#### Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CacheScopeIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

#### Inheritance Hierarchy

xRecord Class Common Table SysCacheFlush Table

#### SysClientAccessLog

### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClientComputer</td>
<td>String</td>
<td>UserIdStr</td>
<td></td>
<td>Name</td>
</tr>
<tr>
<td>createdBy</td>
<td>String</td>
<td>CreatedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td>CreatedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EventsContainer</td>
<td>Container</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Partition</td>
<td>Int64</td>
<td>Partition</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SessionId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Relations

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition</td>
<td>Partitions</td>
</tr>
<tr>
<td>INDEX</td>
<td>ALLOW DUPLICATES</td>
</tr>
<tr>
<td>-------</td>
<td>------------------</td>
</tr>
<tr>
<td>CreatedByIdx</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateDelete. The Application Object Server authorizes each create and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SysClientAccessLog Table

**SysClientSessions**

The SysClientSessions contains the data for the client sessions that are currently active in the system.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>clientComputer</td>
<td>String</td>
<td>UserldStr</td>
<td></td>
<td>Name</td>
</tr>
<tr>
<td>clientType</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DataPartition</td>
<td>String</td>
<td>PartitionKey</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEL_company</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEL_Login_time</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>helpLanguage</td>
<td>String</td>
<td>InstalledLanguageId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LoginDateTime</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ServerId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SessionId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sessionType</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sid</td>
<td>String</td>
<td>Sid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-------------</td>
<td>--------</td>
<td>---------------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Status</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>userId</td>
<td>String</td>
<td>UserId</td>
<td></td>
<td>ID for the user</td>
</tr>
<tr>
<td>userLanguage</td>
<td>String</td>
<td>InstalledLanguageId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataPartition</td>
<td>Partitions</td>
</tr>
<tr>
<td>Relation_SysClientSessions1</td>
<td>SysServerSessions</td>
</tr>
<tr>
<td>Relation_SysClientSessions2</td>
<td>UserInfo</td>
</tr>
<tr>
<td>ServerId</td>
<td>SysServerSessions</td>
</tr>
<tr>
<td>userId</td>
<td>UserInfo</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServerId</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SessionId</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>Yes</td>
<td>Status</td>
</tr>
<tr>
<td>Status_ClientType_UserId</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SysClientSessions Table

**SysConfig**

The SysConfig table contains license and configuration information.

**Fields**
<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>configType</td>
<td>Enum</td>
<td></td>
<td>ConfigType</td>
<td></td>
</tr>
<tr>
<td>createdBy</td>
<td>String</td>
<td></td>
<td>CreatedBy</td>
<td></td>
</tr>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td>CreatedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dEL_CreatedTime</td>
<td>Integer</td>
<td>DEL_CreatedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>dEL_ModifiedTime</td>
<td>Integer</td>
<td>DEL_ModifiedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>expiration</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>modifiedBy</td>
<td>String</td>
<td></td>
<td>ModifiedBy</td>
<td></td>
</tr>
<tr>
<td>modifiedDateTime</td>
<td>UtcDateTime</td>
<td>ModifiedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td></td>
<td>RecId</td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td>RecVersion</td>
<td></td>
</tr>
<tr>
<td>shadowValue</td>
<td>String</td>
<td></td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>timestamp</td>
<td>String</td>
<td></td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>userCount</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Indexes**
Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysConfig Table

SysEncryptionKey

The SysEncryptionKey table stores the encryption key that is used to encrypt the EP query string and post the data parameters.

Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>createdBy</td>
<td>String</td>
<td></td>
<td>CREATEDBY</td>
<td></td>
</tr>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td></td>
<td>CREATEDDATETIME</td>
<td></td>
</tr>
<tr>
<td>dEL_CreatedTime</td>
<td>Integer</td>
<td></td>
<td>DEL_CreatedTime</td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>dEL_ModifiedTime</td>
<td>Integer</td>
<td></td>
<td>DEL_ModifiedTime</td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>Key</td>
<td>Container</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>modifiedBy</td>
<td>String</td>
<td></td>
<td>MODIFIEDBY</td>
<td></td>
</tr>
<tr>
<td>modifiedDateTime</td>
<td>UtcDateTime</td>
<td></td>
<td>MODIFIEDDATETIME</td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td></td>
<td>RECLD</td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td>RECVERSION</td>
<td></td>
</tr>
</tbody>
</table>
### Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecId</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

### Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

### Inheritance Hierarchy

xRecord Class Common Table SysEncryptionKey Table

### SysGlobalConfiguration

The SysGlobalConfiguration table stores system level global setting that can be used to configure specific components.

#### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ServerId</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SettingLevel</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nameldx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

### Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

### Inheritance Hierarchy

xRecord Class Common Table SysGlobalConfiguration Table
SysInheritanceRelations

The SysInheritanceRelations framework helper table for table inheritance. The table stores table inheritance hierarchy related information.

Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>MainTableId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RelatedTableId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RelatedMain</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Inheritance Hierarchy

xRecord Class Common Table SysInheritanceRelations Table

SysLastError

The SysLastError table is storage for the usage data that is recorded as users navigate the system.

Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>company</td>
<td>String</td>
<td>SelectableDataArea</td>
<td></td>
<td>ID for the company you can select</td>
</tr>
<tr>
<td>designName</td>
<td>String</td>
<td>UtilElementName</td>
<td></td>
<td>Name of the application element.</td>
</tr>
<tr>
<td>elementName</td>
<td>String</td>
<td>UtilElementName</td>
<td></td>
<td>Name of the application element.</td>
</tr>
<tr>
<td>isKernel</td>
<td>Enum</td>
<td>boolean</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>Partition</td>
<td>Int64</td>
<td>Partition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-------------</td>
<td>------------</td>
<td>---------------</td>
<td>------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>recordType</td>
<td>Enum</td>
<td></td>
<td>UtilElementType</td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td>RecVersion</td>
<td></td>
</tr>
<tr>
<td>userId</td>
<td>String</td>
<td></td>
<td>UserId</td>
<td>ID for the user</td>
</tr>
<tr>
<td>value</td>
<td>Container</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>isVirtual_Extern</td>
<td>DataArea</td>
</tr>
<tr>
<td>Partition</td>
<td>Partitions</td>
</tr>
<tr>
<td>Relation_SysLastValue1</td>
<td>UserInfo</td>
</tr>
<tr>
<td>Relation_SysLastValue2</td>
<td>DataArea</td>
</tr>
<tr>
<td>userId</td>
<td>UserInfo</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecordType</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>UserId</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Inheritance Hierarchy**

xRecord Class Common Table SysLastValue Table

**SysModel**

The SysModel table contains information about installed models on the system.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>createdBy</td>
<td>String</td>
<td></td>
<td>CreatedBy</td>
<td></td>
</tr>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td></td>
<td>CreatedDateTime</td>
<td></td>
</tr>
<tr>
<td>Layer</td>
<td>Int64</td>
<td></td>
<td>LayerRecid</td>
<td>The ID of the layer.</td>
</tr>
<tr>
<td>modifiedBy</td>
<td>String</td>
<td></td>
<td>ModifiedBy</td>
<td></td>
</tr>
<tr>
<td>modifiedDateTime</td>
<td>UtcDateTime</td>
<td></td>
<td>ModifiedDateTime</td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
<td>---------------</td>
<td>------------------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SysModel Table

**SysModelElement**

The SysModelElement table lists the ModelElements that the installation holds.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AxId</td>
<td>Integer</td>
<td>UtilElementId</td>
<td></td>
<td>Unique internal identification number of the application object.</td>
</tr>
<tr>
<td>ElementType</td>
<td>Int64</td>
<td>ModelElementType</td>
<td></td>
<td>The ID of an ElementType</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Origin</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ParentId</td>
<td>Integer</td>
<td>UtilElementParentId</td>
<td></td>
<td>The unique internal identification number of a parent application object</td>
</tr>
<tr>
<td>ParentModelElement</td>
<td>Int64</td>
<td>ParentModelElement Recid</td>
<td></td>
<td>The ID of a parent model element</td>
</tr>
<tr>
<td>PartOfInheritance</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
<td>---------------</td>
<td>------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RootModelElement</td>
<td>Int64</td>
<td>RootModelElementRecid</td>
<td></td>
<td>The ID of a root model element</td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ElementType</td>
<td>SysModelElementType</td>
</tr>
<tr>
<td>Relation_SysModelElementType</td>
<td>SysModelElementType</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SysModelElementType Table

**SysModelElementData**

The SysModelElementData table provides the Layer specific data for any SysModelElement.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>createdBy</td>
<td>String</td>
<td>CreatedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td>CreatedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Layer</td>
<td>Int64</td>
<td>LayerRecid</td>
<td></td>
<td>The ID of the layer.</td>
</tr>
<tr>
<td>LegacyId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ModelElement</td>
<td>Int64</td>
<td>ModelElementRecid</td>
<td></td>
<td>The ID of a ModelElement</td>
</tr>
<tr>
<td>ModelId</td>
<td>Integer</td>
<td>ModelId</td>
<td></td>
<td>The ID of the model.</td>
</tr>
</tbody>
</table>
FIELD | TYPE | EXTENDED TYPE | ENUMERATION TYPE | DESCRIPTION
--- | --- | --- | --- | ---
modifiedBy | String | ModifiedBy | | 
modifiedDateTime | UtcDateTime | ModifiedDateTime | | 
Recl | Int64 | Recl | | 
recVersion | Integer | RecVersion | | 
SaveCount | Int | | | 

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer</td>
<td>SysModelLayer</td>
</tr>
<tr>
<td>ModelElement</td>
<td>SysModelElement</td>
</tr>
<tr>
<td>ModelId</td>
<td>SysModel</td>
</tr>
<tr>
<td>Relation_SysModel</td>
<td>SysModel</td>
</tr>
<tr>
<td>Relation_SysModelElement</td>
<td>SysModelElement</td>
</tr>
<tr>
<td>Relation_SysModelLayer</td>
<td>SysModelLayer</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReclIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SysModelElementData Table

**SysModelElementDataOld**

The SysModelElementDataOld table provides the Layer specific data for any SysModelElementOld.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>createdBy</td>
<td>String</td>
<td>CreatedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------</td>
<td>---------------</td>
<td>------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td>CreatedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Layer</td>
<td>Int64</td>
<td>LayerRecid</td>
<td></td>
<td>The ID of the layer.</td>
</tr>
<tr>
<td>LegacyId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ModelElement</td>
<td>Int64</td>
<td>ModelElementRecid</td>
<td></td>
<td>The ID of a ModelElement</td>
</tr>
<tr>
<td>ModelId</td>
<td>Integer</td>
<td>ModelId</td>
<td></td>
<td>The ID of the model.</td>
</tr>
<tr>
<td>modifiedBy</td>
<td>String</td>
<td>ModifiedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>modifiedDateTime</td>
<td>UtcDateTime</td>
<td>ModifiedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReclId</td>
<td>Int64</td>
<td>ReclId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SaveCount</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer</td>
<td>SysModelLayerOld</td>
</tr>
<tr>
<td>ModelElement</td>
<td>SysModelElementOld</td>
</tr>
<tr>
<td>ModelId</td>
<td>SysModelOld</td>
</tr>
<tr>
<td>Relation_SysModelElementOld</td>
<td>SysModelElementOld</td>
</tr>
<tr>
<td>Relation_SysModelLayerOld</td>
<td>SysModelLayerOld</td>
</tr>
<tr>
<td>Relation_SysModelOld</td>
<td>SysModelOld</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReclIdIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**
SysModelElementLabel

The SysModelElementLabel table contains the label text for a given language.

Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Id</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LabelId</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Module</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td></td>
<td>RecId</td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td>RecVersion</td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ModuleLangIDIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RecIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysModelElementLabel Table

SysModelElementLabelOld

The SysModelElementLabelOld table contains the label text for a given language.

Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>---------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Id</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LabelId</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Module</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SysModelElementLabelOld Table

**SysModelElementOld**

The SysModelElementOld table lists the ModelElements that the installation holds.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AxId</td>
<td>Integer</td>
<td>UtilElementId</td>
<td></td>
<td>Unique internal identification number of the application object.</td>
</tr>
<tr>
<td>ElementType</td>
<td>Int64</td>
<td>ModelElementType</td>
<td></td>
<td>The ID of an ElementType</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Origin</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Relations

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ElementType</td>
<td>SysModelElementTypeOld</td>
</tr>
<tr>
<td>Relation_SysModelElementTypeOld</td>
<td>SysModelElementTypeOld</td>
</tr>
</tbody>
</table>

### Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIDIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

### Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

### Inheritance Hierarchy

xRecord Class Common Table SysModelElementOld Table

### SysModelElementSource

The SysModelElementSource table contains the Source Text for all SysModelElements that have source.

### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer</td>
<td>Int64</td>
<td>LayerRecid</td>
<td></td>
<td>The ID of the layer.</td>
</tr>
</tbody>
</table>
### FIELD | TYPE | EXTENDED TYPE | ENUMERATION TYPE | DESCRIPTION
--- | --- | --- | --- | ---
ModelElement | Int64 | ModelElementRecid |  | The ID of a ModelElement
RecId | Int64 | RecId |  | 
recVersion | Integer | RecVersion |  | 
Source | Container | |  | 

#### Relations

**RELATION**

**TABLE**

**Layer**

SysModelElementData

**Relation_SysModelElementData**

SysModelElementData

#### Indexes

**INDEX**

**ALLOW DUPLICATES**

**FIELDS**

RecIdIdx | No |  |

#### Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

#### Inheritance Hierarchy

xRecord Class Common Table SysModelElementSource Table

### SysModelElementSourceOld

The SysModelElementSourceOld table contains the Source Text for all SysModelElementsOld that have source.

#### Fields

**FIELD**

**TYPE**

**EXTENDED TYPE**

**ENUMERATION TYPE**

**DESCRIPTION**

Layer | Int64 | LayerRecid |  | The ID of the layer.

ModelElement | Int64 | ModelElementRecid |  | The ID of a ModelElement

RecId | Int64 | RecId |  | 

recVersion | Integer | RecVersion |  | 

Source | Container |  |  | 

---
### Relations

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer</td>
<td>SysModelElementDataOld</td>
</tr>
<tr>
<td>Relation_SysModelElementDataOld</td>
<td>SysModelElementDataOld</td>
</tr>
</tbody>
</table>

### Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

### Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

### Inheritance Hierarchy

xRecord Class Common Table SysModelElementSourceOld Table

### SysModelElementType

The SysModelElementType table specifies the possible SysModelElement types. Its Recid is backwards compatible with the UtilRecordType enum for the 'old' element types.

### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>ModelElementTypeName</td>
<td></td>
<td>The name of the element type.</td>
</tr>
<tr>
<td>ParentType</td>
<td>Int64</td>
<td>ModelElementType</td>
<td></td>
<td>The ID of an ElementType</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TreeNodeName</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Relations

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>SysModelElementType</td>
</tr>
</tbody>
</table>

### Indexes
Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SysModelElementType Table

SysModelElementTypeOld

The SysModelElementTypeOld table specifies the possible SysModelElementOld types. Its Recid is backwards compatible with the UtilRecordType enum for the 'old' element types.

Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>ModelElementTypeNname</td>
<td></td>
<td>The name of the element type.</td>
</tr>
<tr>
<td>ParentType</td>
<td>Int64</td>
<td>ModelElementType</td>
<td></td>
<td>The ID of an ElementType</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TreeNodeName</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Relations

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>SysModelElementType</td>
</tr>
</tbody>
</table>

Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not
authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SysModelElementTypeOld Table

**SysModelLayer**

The SysModelLayer table lists the possible LayerId and Name. If Model data exists in a layer it reports the aggregated version number for that layer.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsDirty</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Layer</td>
<td>Enum</td>
<td></td>
<td>UtilEntryLevel</td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td></td>
<td>RecId</td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td>RecVersion</td>
<td></td>
</tr>
<tr>
<td>VersionNumber</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SysModelLayer Table

**SysModelLayerOld**

The SysModelLayerOld table lists the possible LayerId and Name. If Model data exists in a layer it reports the aggregated version number for that layer.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer</td>
<td>Enum</td>
<td></td>
<td>UtilEntryLevel</td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td></td>
<td>RecId</td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td>RecVersion</td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-------------</td>
<td>-------</td>
<td>---------------------</td>
<td>------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>VersionNumber</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SysModelLayerOld Table

**SysModelManifest**

The SysModelManifest table contains the manifest information about deployed models, such as Description, Publisher and Version of a model.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Integer</td>
<td>ModelManifestCategoryRecId</td>
<td></td>
<td>The ID of the model category</td>
</tr>
<tr>
<td>Description</td>
<td>String</td>
<td>ModelDescription</td>
<td></td>
<td>The description of the model</td>
</tr>
<tr>
<td>DisplayName</td>
<td>String</td>
<td>ModelDisplayName</td>
<td></td>
<td>The display name of the model</td>
</tr>
<tr>
<td>Model</td>
<td>Int64</td>
<td>ModelRecid</td>
<td></td>
<td>The ID of the model</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>ModelName</td>
<td></td>
<td>The name of the model in the model store</td>
</tr>
<tr>
<td>Publisher</td>
<td>String</td>
<td>ModelPublisher</td>
<td></td>
<td>The publisher of the model</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signed</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VersionBuildNo</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Field Groups**

<table>
<thead>
<tr>
<th>FIELD GROUP</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoIdentification</td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>SysModelManifestCategory</td>
</tr>
<tr>
<td>Model</td>
<td>SysModel</td>
</tr>
<tr>
<td>Relation_SysModel</td>
<td>SysModel</td>
</tr>
<tr>
<td>Relation_SysModelManifestCategory</td>
<td>SysModelManifestCategory</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ModelNameIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RecIDIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SysModelManifest Table

**SysModelManifestCategory**

The SysModelManifestCategory table contains the category aspect of the manifest information for deployed models.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>VersionMajor</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VersionMinor</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VersionRevision</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SysModelManifestCategory Table

**SysModelManifestCategoryOld**

The SysModelManifestCategoryOld table contains the category aspect of the manifest information for deployed models.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>ModelManifestCategoryName</td>
<td></td>
<td>The name of the model category.</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nameldx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.
Inheritance Hierarchy

xRecord Class Common Table SysModelManifestCategoryOld Table

SysModelManifestOld

The SysModelManifestOld table contains the manifest information about deployed models, such as Description, Publisher and Version of a model.

### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Integer</td>
<td>ModelManifestCategoryRecId</td>
<td></td>
<td>The ID of the model category</td>
</tr>
<tr>
<td>Description</td>
<td>String</td>
<td>ModelDescription</td>
<td></td>
<td>The description of the model.</td>
</tr>
<tr>
<td>DisplayName</td>
<td>String</td>
<td>ModelDisplayName</td>
<td></td>
<td>The display name of the model.</td>
</tr>
<tr>
<td>Model</td>
<td>Int64</td>
<td>ModelRecidOld</td>
<td></td>
<td>The ID of the model (old).</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>ModelName</td>
<td></td>
<td>The name of the model in the model store.</td>
</tr>
<tr>
<td>Publisher</td>
<td>String</td>
<td>ModelPublisher</td>
<td></td>
<td>The publisher of the model.</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>Recld</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signed</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VersionBuildNo</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VersionMajor</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VersionMinor</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VersionRevision</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Field Groups

<table>
<thead>
<tr>
<th>FIELD GROUP</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autoidentification</td>
<td></td>
</tr>
</tbody>
</table>

### Relations
### Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ModelNameIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

### Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

### Inheritance Hierarchy

xRecord Class Common Table SysModelManifestOld Table

### SysModelOld

The SysModelOld table contains information about installed models on the system.

#### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>createdBy</td>
<td>String</td>
<td>CreatedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td>CreatedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Layer</td>
<td>Int64</td>
<td>LayerRecid</td>
<td></td>
<td>The ID of the layer.</td>
</tr>
<tr>
<td>modifiedBy</td>
<td>String</td>
<td>ModifiedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>modifiedDateTime</td>
<td>UtcDateTime</td>
<td>ModifiedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>State</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Indexes
### Security Note
Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

### Inheritance Hierarchy

**SysOccConfiguration**

The SysOccConfiguration table stores the global concurrency model setting and updates the conflict exception login policy.

### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoUpdateRecVersion</td>
<td>Enum</td>
<td>boolean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GlobalOccMode</td>
<td>Enum</td>
<td>GlobalOccMode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LogHandledUpdateConflicts</td>
<td>Enum</td>
<td>boolean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>ReclId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UniqueIndex</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UseReadUncommittedForAll</td>
<td>Enum</td>
<td>boolean</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW Duplicates</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>UniqueIndex</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
SysRecordLevelSecurity

The SysRecordLevelSecurity table contains all the record level security restrictions that are configured by the system administrator. The restrictions are persisted on a per company, per group basis.

### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_unused</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td>ID for the company you can select</td>
</tr>
<tr>
<td>companyId</td>
<td>String</td>
<td>SelectableDataArea</td>
<td></td>
<td>ID for the company</td>
</tr>
<tr>
<td>createdBy</td>
<td>String</td>
<td>CreatedBy</td>
<td></td>
<td>ID for the company</td>
</tr>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td>CreatedDateTime</td>
<td></td>
<td>ID for the company</td>
</tr>
<tr>
<td>dEL_CreatedTime</td>
<td>Integer</td>
<td>DEL_CreatedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>DEL_groupId</td>
<td>String</td>
<td>UserGroupId</td>
<td></td>
<td>ID for the user group</td>
</tr>
<tr>
<td>dEL_ModifiedTime</td>
<td>Integer</td>
<td>DEL_ModifiedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>modifiedBy</td>
<td>String</td>
<td>ModifiedBy</td>
<td></td>
<td>ID for the company</td>
</tr>
<tr>
<td>modifiedDateTime</td>
<td>UtcDateTime</td>
<td>ModifiedDateTime</td>
<td></td>
<td>ID for the company</td>
</tr>
<tr>
<td>Partition</td>
<td>Int64</td>
<td>Partition</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>restriction</td>
<td>Container</td>
<td></td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------------</td>
<td>--------</td>
<td>---------------</td>
<td>------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>SecurityRole</td>
<td>Int64</td>
<td>ReclId</td>
<td></td>
<td>Name of the security role</td>
</tr>
<tr>
<td>tabId</td>
<td>Integer</td>
<td>TableId</td>
<td></td>
<td>ID for the table</td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>companyId</td>
<td>DataArea</td>
</tr>
<tr>
<td>DEL_groupId</td>
<td>UserGroupInfo</td>
</tr>
<tr>
<td>Partition</td>
<td>Partitions</td>
</tr>
<tr>
<td>Relation_SecurityRole</td>
<td>SecurityRole</td>
</tr>
<tr>
<td>SecurityRole</td>
<td>SecurityRole</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecId</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Role</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SysRecordLevelSecurity Table

**SysServerSessions**

The SysServerSessions table is used to store information about the active AOS Servers in the system.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AOSAccount</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AOSId</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEL_LastUpdateTime</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------</td>
<td>---------------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>DEL_Login_time</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instance_Name</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LastUpdateDateTime</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LoadBalance</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LoginDateTime</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ServerId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Workload</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoadBalance</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>ServerId</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table SysServerSessions Table

**SysSetbasedHelper**

The SysSetbasedHelper framework helper table for table inheritance set-based operations.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CandidateRecId</td>
<td>Int64</td>
<td>Recl</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>---------------</td>
<td>------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td></td>
<td>RecId</td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td>RecVersion</td>
<td></td>
</tr>
</tbody>
</table>

Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CandidateRecIdIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Inheritance Hierarchy

xRecord Class Common Table SysSetbasedHelper Table

SystemSequences

The SystemSequences table holds the next available record ID block for each table.

Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>cycle</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>dataAreaId</td>
<td>String</td>
<td>DataAreaId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxVal</td>
<td>Int64</td>
<td>Recld</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>minVal</td>
<td>Int64</td>
<td>Recld</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nextVal</td>
<td>Int64</td>
<td>Recld</td>
<td></td>
<td>Unique ID for the record in the database</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>Recld</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tabId</td>
<td>Integer</td>
<td>TableId</td>
<td></td>
<td>ID for the table</td>
</tr>
</tbody>
</table>

Relations
Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table SystemSequences Table

TableCollectionList

The TableCollectionList table stores the mapping between table collections and virtual companies.

Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition</td>
<td>Int64</td>
<td>Partition</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>ReclId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tableCollection</td>
<td>String</td>
<td>UtilElementName</td>
<td></td>
<td>Name of the application element.</td>
</tr>
<tr>
<td>virtualDataArea</td>
<td>String</td>
<td>VirtualDataArea</td>
<td></td>
<td>ID for a virtual company</td>
</tr>
</tbody>
</table>

Relations

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>isVirtual_Extern</td>
<td>DataArea</td>
</tr>
<tr>
<td>parentId_Extern</td>
<td>UtilElements</td>
</tr>
<tr>
<td>Partition</td>
<td>Partitions</td>
</tr>
</tbody>
</table>
**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>VirtualDataArea</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table TableCollectionList Table

**TimeZonesList**

The TimeZonesList table contains the list of the time zones that are supported.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnumName</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EnumPosition</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td></td>
<td>RecId</td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td>RecVersion</td>
<td></td>
</tr>
<tr>
<td>TimeZoneKeyName</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TzEnum</td>
<td>Enum</td>
<td></td>
<td>Timezone</td>
<td></td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnumPosition</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>TzEnum</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.
Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy
xRecord Class Common Table TimeZonesList Table

TimeZonesRulesData
The TimeZonesRulesData table contains the GMT offsets and daylight saving time information for all time zones that are supported.

Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bias</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBias</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDay</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDayOfWeek</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DHour</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMinute</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMonth</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSecond</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DYear</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RuleId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBias</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDay</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDayOfWeek</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHour</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMinute</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMonth</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSecond</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>---------------</td>
<td>------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>SYear</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TzEnum</td>
<td>Enum</td>
<td></td>
<td>Timezone</td>
<td></td>
</tr>
<tr>
<td>Year</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

**RELATION**

**TABLE**

<table>
<thead>
<tr>
<th>Relation_TimeZonesRulesData1</th>
<th>TimeZonesList</th>
</tr>
</thead>
<tbody>
<tr>
<td>TzEnum</td>
<td>TimeZonesList</td>
</tr>
</tbody>
</table>

**Indexes**

**INDEX**

**ALLOW DUPLICATES**

**FIELDS**

<table>
<thead>
<tr>
<th>RuleId</th>
<th>No</th>
</tr>
</thead>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table TimeZonesRulesData Table

**UserDataAreaFilter**

The UserDataAreaFilter table contains a list of selectable companies for a user. It is populated by invoking the populateSelectaTableCompanies method on the SecurityRights class.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataArea</td>
<td>String</td>
<td>DataAreaId</td>
<td></td>
<td>ID for an area of data</td>
</tr>
<tr>
<td>Partition</td>
<td>Int64</td>
<td>Partition</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### User

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>String</td>
<td>UserId</td>
<td></td>
<td>ID for the user</td>
</tr>
</tbody>
</table>

#### Relations

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataArea</td>
<td>DataArea</td>
</tr>
<tr>
<td>Partition</td>
<td>Partitions</td>
</tr>
<tr>
<td>Relation_DataArea</td>
<td>DataArea</td>
</tr>
<tr>
<td>Relation_User</td>
<td>UserInfo</td>
</tr>
<tr>
<td>User</td>
<td>UserInfo</td>
</tr>
</tbody>
</table>

#### Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataAreaIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

#### Inheritance Hierarchy

xRecord Class  Common Table  UserDataAreaFilter Table

### UserInfo

The UserInfo table contains a list of users and their active directory and default information.

#### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>accountType</td>
<td>Enum</td>
<td></td>
<td>UserAccountType</td>
<td></td>
</tr>
<tr>
<td>autoInfo</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>autoLogOff</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>autoUpdate</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>clientAccessLogLevel</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>company</td>
<td>String</td>
<td>SelectableDataArea</td>
<td></td>
<td>ID for the company you can select</td>
</tr>
<tr>
<td>compilerWarningLevel</td>
<td>Enum</td>
<td></td>
<td>CompilerWarningLevel</td>
<td></td>
</tr>
<tr>
<td>confirmDelete</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------</td>
<td>---------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>confirmUpdate</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>credentialRecId</td>
<td>int64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>debuggerPopup</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>debugInfo</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>defaultPartition</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>DEL__unused1</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEL__unused2</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEL_defaultModelId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEL_osAccountName</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEL_password</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEL_startupMenu</td>
<td>String</td>
<td>UtilElementName</td>
<td></td>
<td>Name of the application element.</td>
</tr>
<tr>
<td>enable</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>enabledOnce</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>externalId</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>externalIdType</td>
<td>Enum</td>
<td></td>
<td>ExternalIdType</td>
<td></td>
</tr>
<tr>
<td>externalUser</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filterByGridOnByDefa ult</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>formFontName</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>formFontSize</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>garbagecollectlimit</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>generalInfo</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------------</td>
<td>---------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>globalExcelExportFile Path</td>
<td>String</td>
<td></td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>globalExcelExportLocation</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>globalExcelExportMode</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>globalFormOpenMode</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>globalListPageLinkMode</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>helplanguage</td>
<td>String</td>
<td></td>
<td>InstalledLanguageId</td>
<td></td>
</tr>
<tr>
<td>historyLimit</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>homePageRefreshDuration</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>String</td>
<td>UserId</td>
<td></td>
<td>ID for the user</td>
</tr>
<tr>
<td>IdentityProvider</td>
<td>String</td>
<td></td>
<td>NetworkDomain</td>
<td></td>
</tr>
<tr>
<td>infologLevel</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>issuerRecId</td>
<td>int64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>language</td>
<td>String</td>
<td></td>
<td>InstalledLanguageId</td>
<td></td>
</tr>
<tr>
<td>messageLimit</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>UserIdStr</td>
<td></td>
<td>Name</td>
</tr>
<tr>
<td>networkAlias</td>
<td>String</td>
<td></td>
<td>NetworkAlias</td>
<td></td>
</tr>
<tr>
<td>networkDomain</td>
<td>String</td>
<td></td>
<td>NetworkDomain</td>
<td></td>
</tr>
<tr>
<td>notifyTimeZoneMismatch</td>
<td>Enum</td>
<td></td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------</td>
<td>---------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Partition</td>
<td>Int64</td>
<td>Partition</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>preferredCalendar</td>
<td>Enum</td>
<td></td>
<td>PreferredCalendar</td>
<td></td>
</tr>
<tr>
<td>PreferredLocale</td>
<td>String</td>
<td>PreferredLocale</td>
<td></td>
<td></td>
</tr>
<tr>
<td>preferredTimeZone</td>
<td>Enum</td>
<td>Timezone</td>
<td></td>
<td></td>
</tr>
<tr>
<td>propertyFontName</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>propertyFontSize</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>querytimeLimit</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>Recld</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>reportBottomMargin</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reportFontName</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reportFontSize</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reportLeftMargin</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reportRightMargin</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reportTopMargin</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>showAOTLayer</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>showModelNameInAOT</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>showStatusLine</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>showToolbar</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sid</td>
<td>String</td>
<td>Sid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>startupProject</td>
<td>String</td>
<td>UtilElementName</td>
<td>Name of the application element.</td>
<td></td>
</tr>
<tr>
<td>statuslinelInfo</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Field Groups

<table>
<thead>
<tr>
<th>FIELD GROUP</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoLookup</td>
<td>id, accountType, name, networkAlias, networkDomain, enable</td>
</tr>
<tr>
<td>AutoReport</td>
<td></td>
</tr>
</tbody>
</table>

Relations

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>UserInfo</td>
</tr>
<tr>
<td>isVirtual_Extern</td>
<td>DataArea</td>
</tr>
<tr>
<td>Partition</td>
<td>Partitions</td>
</tr>
<tr>
<td>Relation_UserInfo</td>
<td>DataArea</td>
</tr>
</tbody>
</table>

Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>IdOnly</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Sid</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SidOnly</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

Security Note

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateDelete. The Application Object Server authorizes each create and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

Inheritance Hierarchy

xRecord Class Common Table UserInfo Table

UserInfoStartupModel

The UserInfoStartupModel table holds the preferred startup model for each layer for each user.

Fields
<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer</td>
<td>Enum</td>
<td></td>
<td>UtilEntryLevel</td>
<td></td>
</tr>
<tr>
<td>Modelld</td>
<td>Int64</td>
<td>ModelRecid</td>
<td></td>
<td>The ID of the model.</td>
</tr>
<tr>
<td>Partition</td>
<td>Int64</td>
<td>Partition</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>Recld</td>
<td>Int64</td>
<td>Recld</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UserId</td>
<td>String</td>
<td>UserGroupId</td>
<td></td>
<td>ID for the user group</td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modelld</td>
<td>SysModelManifest</td>
</tr>
<tr>
<td>Partition</td>
<td>Partitions</td>
</tr>
<tr>
<td>Relation_SysModelManifest</td>
<td>SysModelManifest</td>
</tr>
<tr>
<td>Relation_UserInfo</td>
<td>UserInfo</td>
</tr>
<tr>
<td>Relation_UserInfoStartupModel3</td>
<td>DEL_UserGroupInfo</td>
</tr>
<tr>
<td>UserId</td>
<td>UserInfo</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecIdIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>UserID_Layer_Idx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table UserInfoStartupModel Table
UtilElements

The UtilElements table contains the application that is shown in the AOT.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseVersion</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>code</td>
<td>Container</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>createdBy</td>
<td>String</td>
<td></td>
<td>CreatedBy</td>
<td></td>
</tr>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td></td>
<td>CreatedDateTime</td>
<td></td>
</tr>
<tr>
<td>dEL_CreatedTime</td>
<td>Integer</td>
<td>DEL_CreatedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>dEL_ModifiedTime</td>
<td>Integer</td>
<td>DEL_ModifiedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>modifiedBy</td>
<td>String</td>
<td></td>
<td>ModifiedBy</td>
<td></td>
</tr>
<tr>
<td>modifiedDateTime</td>
<td>UtcDateTime</td>
<td></td>
<td>ModifiedDateTime</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td></td>
<td>UtilElementName</td>
<td>Name of the application element.</td>
</tr>
<tr>
<td>parentId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td></td>
<td>RecId</td>
<td></td>
</tr>
<tr>
<td>recordType</td>
<td>Enum</td>
<td></td>
<td>UtilElementType</td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td>RecVersion</td>
<td></td>
</tr>
<tr>
<td>saveCount</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>source</td>
<td>Container</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>utilLevel</td>
<td>Enum</td>
<td></td>
<td>UtilEntryLevel</td>
<td></td>
</tr>
<tr>
<td>version</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Indexes**
<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>No</td>
<td>recordType, parentId, name, utilLevel</td>
</tr>
<tr>
<td>RecId</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table UtilElements Table

**UtilElementsOld**

The UtilElementsOld table contains the application model stored in the application folder. It is used during the upgrade process.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseVersion</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>code</td>
<td>Container</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>createdBy</td>
<td>String</td>
<td>CreatedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td>CreatedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dEL_CreatedTime</td>
<td>Integer</td>
<td>DEL_CreatedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>dEL_ModifiedTime</td>
<td>Integer</td>
<td>DEL_ModifiedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>modifiedBy</td>
<td>String</td>
<td>ModifiedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>modifiedDateTime</td>
<td>UtcDateTime</td>
<td>ModifiedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>UtilElementName</td>
<td></td>
<td>Name of the application element.</td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>---------------</td>
<td>------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>parentId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>Recld</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recType</td>
<td>Enum</td>
<td></td>
<td>UtilElementType</td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>saveCount</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>source</td>
<td>Container</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>utilLevel</td>
<td>Enum</td>
<td></td>
<td>UtilEntryLevel</td>
<td></td>
</tr>
<tr>
<td>version</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>No</td>
<td>recordType, parentId, name, utilLevel</td>
</tr>
<tr>
<td>RecId</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Inheritance Hierarchy

xRecord Class Common Table UtilElementsOld Table

UtilIdElements

The UtilIdElements table contains the application model shown in the AOT.

Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseVersion</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>code</td>
<td>Container</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>createdBy</td>
<td>String</td>
<td>CreatedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td>CreatedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dEL_CreatedTime</td>
<td>Integer</td>
<td>DEL_CreatedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------------</td>
<td>------------</td>
<td>---------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>dEL_ModifiedTime</td>
<td>Integer</td>
<td>DEL_ModifiedTime</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>id</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>modifiedBy</td>
<td>String</td>
<td></td>
<td>ModBy</td>
<td></td>
</tr>
<tr>
<td>modifiedDateTime</td>
<td>UtcDateTime</td>
<td></td>
<td>ModifiedDateTime</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td></td>
<td>UtilElementName</td>
<td>Name of the application element.</td>
</tr>
<tr>
<td>parentId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td></td>
<td>RecId</td>
<td></td>
</tr>
<tr>
<td>recordType</td>
<td>Enum</td>
<td></td>
<td>UtilElementType</td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td></td>
<td>RecVersion</td>
<td></td>
</tr>
<tr>
<td>saveCount</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>utilLevel</td>
<td>Enum</td>
<td></td>
<td>UtilEntryLevel</td>
<td></td>
</tr>
<tr>
<td>version</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>No</td>
<td>recordType, parentId, name, utilLevel</td>
</tr>
<tr>
<td>RecId</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table UtilIdElements Table
### UtilIdElementsOld

The `UtilIdElementsOld` table contains the application model stored in the application folder. It is used during the upgrade process.

#### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseVersion</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>code</td>
<td>Container</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>createdBy</td>
<td>String</td>
<td>CreatedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>createdDateTime</td>
<td>UtcDateTime</td>
<td>CreatedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dEL_CreatedTime</td>
<td>Integer</td>
<td>DEL_CreatedTime</td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
<td></td>
</tr>
<tr>
<td>dEL_ModifiedTime</td>
<td>Integer</td>
<td>DEL_ModifiedTime</td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>modifiedBy</td>
<td>String</td>
<td>ModifiedBy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>modifiedDateTime</td>
<td>UtcDateTime</td>
<td>ModifiedDateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>UtilElementName</td>
<td>Name of the application element.</td>
<td></td>
</tr>
<tr>
<td>parentId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReclId</td>
<td>Int64</td>
<td>ReclId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordType</td>
<td>Enum</td>
<td></td>
<td>UtilElementType</td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>saveCount</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>utilLevel</td>
<td>Enum</td>
<td>UtilEntryLevel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>version</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Indexes

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>No</td>
<td>recordType, parentId, name, utilLevel</td>
</tr>
<tr>
<td>name</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Recid</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

### Inheritance Hierarchy

xRecord Class Common Table UtilIdElementsOld Table

### UtilModels

The UtilModels table contains information about models that are installed on the system.

### Fields

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CategoryId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>String</td>
<td>ModelDescription</td>
<td></td>
<td>The description of the model.</td>
</tr>
<tr>
<td>DisplayName</td>
<td>String</td>
<td>ModelDisplayName</td>
<td></td>
<td>The display name of the model.</td>
</tr>
<tr>
<td>Id</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>InstallMode</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Layer</td>
<td>Enum</td>
<td>UtilEntryLevel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MarkedForRemoval</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ModelGroupId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>ModelName</td>
<td></td>
<td>The name of the model in the model store.</td>
</tr>
<tr>
<td>Publisher</td>
<td>String</td>
<td>ModelPublisher</td>
<td></td>
<td>The publisher of the model.</td>
</tr>
<tr>
<td>Recid</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signed</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State</td>
<td>String</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------</td>
<td>---------------</td>
<td>------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>VersionBuildNo</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VersionMajor</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VersionMinor</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VersionRevision</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ModelIDIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ModelNameIdx</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RecIDIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table UtilModels Table

**VirtualDataAreaList**

The VirtualDataAreaList table stores the mapping between real companies and virtual companies.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>SelectableDataArea</td>
<td></td>
<td>ID for the company you can select</td>
</tr>
<tr>
<td>Partition</td>
<td>Int64</td>
<td>Partition</td>
<td></td>
<td>(This field applies only to the following version(s): Microsoft Dynamics AX 2012 R3, Microsoft Dynamics AX 2012 R2 (SYS))</td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------</td>
<td>---------</td>
<td>----------------</td>
<td>------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>virtualDataArea</td>
<td>String</td>
<td>VirtualDataArea</td>
<td></td>
<td>ID for a virtual company</td>
</tr>
</tbody>
</table>

**Field Groups**

<table>
<thead>
<tr>
<th>FIELD GROUP</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtualDataAreaRelation</td>
<td></td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>RELATION</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>isVirtual_Extern</td>
<td>DataArea</td>
</tr>
<tr>
<td>Partition</td>
<td>Partitions</td>
</tr>
<tr>
<td>Relation_DataAreaList1</td>
<td>DataArea</td>
</tr>
<tr>
<td>Relation_DataAreaList2</td>
<td>DataArea</td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table VirtualDataAreaList Table

**VSAssembly**

The VSAssembly table contains synchronization information that describes the last time an assembly that is stored under the Visual Studio Projects node in the AOT was deployed.

**Fields**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>TYPE</th>
<th>EXTENDED TYPE</th>
<th>ENUMERATION TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeployTo</td>
<td>Enum</td>
<td></td>
<td>DeployTo</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>AssemblyName</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>TYPE</td>
<td>EXTENDED TYPE</td>
<td>ENUMERATION TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------------</td>
<td>------------</td>
<td>---------------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>ProjectName</td>
<td>String</td>
<td>ProjectName</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProjectType</td>
<td>String</td>
<td>ProjectType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecId</td>
<td>Int64</td>
<td>RecId</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recVersion</td>
<td>Integer</td>
<td>RecVersion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ServerId</td>
<td>Int</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UpdatedDate</td>
<td>UtcDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Indexes**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ALLOW DUPLICATES</th>
<th>FIELDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NameIdx</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

**Security Note**

Use of this table could lead to an Elevation of Privileges attack or a Denial of Service attack. Therefore, the AOSAuthorization property is set to an enumeration value of CreateUpdateDelete. The Application Object Server authorizes each create, update, and delete action on the table by confirming that the current user has permission to perform the requested operation on that table. If the user who initiates the operation is not authorized to perform the operation, an exception occurs.

**Inheritance Hierarchy**

xRecord Class Common Table VSAssembly Table
Dynamics 365 Finance, Supply Chain, and Commerce are extensively customized by partners, value added resellers (VARs), and even some customers. The ability to customize the product is a strength that has historically been supported through overlayering of the application code. The move to the cloud, together with more agile servicing and frequent updates, requires a less intrusive customization model, so that updates are less likely to affect custom solutions. This new model is called extensibility and has replaced customization through overlayering.

Extensibility is the only customization framework in Finance, Supply Chain, and Commerce. Overlayering isn't supported.

Introduction

These introductory topics contain general information about customization. This information includes information about when the transition occurs from customization through overlayering to a purely extension-based model. These topics also explain how to log extensibility requests to Microsoft, and provide answers to frequently asked questions (FAQ).

- Application extensibility plans
- Extensibility requests
- Extensibility FAQ

What's new

Read What's new or changed for extensibility for extensibility-related updates that have been made since July 2017.

Getting started

The topics in this section will help you start to build extensions. They will also help you migrate solutions that are currently based on overlayered code to extension-based solutions. This section includes hands-on labs that walk you through simple customizations.

- Migrate from overlayering to extensions
- Customize model elements through extension
- Customize through extension and overlayering

Fundamentals on extensions

This section includes fundamentals, principles, and practices for making extensions. The guiding principles in these topics discuss how customization must be approached through extensions. These principles include naming guidelines. Additionally, these topics discuss the foundation framework, such as extensions and chain of command.

- Intrusive customizations
- Class extension model in X++
- Class extension - Method wrapping and Chain of Command
- Naming guidelines for extensions
• Relax model restrictions to refactor overlayering into extensions

**How do I create extensions?**

This section includes "How do I?" topics that explain how to customize specific object types or code. Most of these topics are brief and to the point. Because there are many topics here, it might be practical to search for a specific topic.

**Data types**
- Add values to enums through extension
- Modify extended data types (EDTs) through extension

**Classes**
- Register subclasses for factory methods
- Respond by using EventHandlerResult
- Extend the RunBase class
- Customize application startup by using delegates

**Tables**
- Modify existing fields in a table through extension
- Add fields to tables through extension
- Add indexes to tables through extension
- Add relations to tables through extension
- Modify table properties through extension
- Add methods to tables through extension
- Perform business actions throughout the lifecycle of table records

**Forms**
- Add a new data source to a form
- Change the captions of forms through extension
- Modify the properties of form controls through extension

**Others**
- Extending decimal point precision for selected data types
- Add new inventory dimensions through extension

**Reports**
- Extend the list of Electronic reporting (ER) functions
- Customize App Suite reports by using extensions

**Blog posts**
Information about customization is also shared through blogs where various topics are discussed. This section includes reference to some of these blogs.

- Extending Dynamics 365 for Finance and Operations
- Extension methods
- Extensible base enumerations
- Static event subscription
- Subscribing to onValidatingWrite
- Embrace the extensions mindset with Dynamics 365 for Finance and Operations
- Extensible X++ - Method Signatures
How do I create an extensible solution?

This section includes some best practices on how to create/make your solution extensible, so that consumers of your code can extend your solution.

- **Write extensible code**
- **Classes**
- **Methods**
- **Forms**
- **Extended data types**
- **Extensible enums**
- **Delegates**
- **Tables**
- **Attributes that make methods extensible**

**Breaking changes**

When you make your solution extensible, you also help guarantee that you won’t break those extension points later.

- For pointers that can help you avoid breaking your consumers, see [Breaking changes](#).
- The compatibility checker tool can detect metadata breaking changes against a given baseline release or update, helping to ensure backward compatibility.
Reducing implementation and upgrade effort is a major initiative for the development team. The benefits of this initiative are to enable you to quickly take advantage of new innovations from Microsoft and your partners, reduce the total cost of ownership, and improve quality. A major part of this initiative is to change the customization approach for the product. In Dynamics AX 2012, several extension capabilities were added to the product. For example, the ability to do event-based customization using methods pre-and post-events was introduced. Extension capabilities have continued to grow in the evolution to the new application.

Extension-based customizations have several advantages over the legacy approach of overlayering-based customizations, especially when it comes to reducing implementation and upgrade effort.

- Overlayering-based customizations require code upgrade, recompile time, and extensive testing. This limits the ability to seamlessly apply hot fixes. These costs can be an inhibitor for customers to upgrade to newer versions containing innovations from Microsoft and partners.
- Extension-based customizations also improve the development experience. Models containing overlayered customizations must be in the same package as the base objects. This results in longer compile cycles and larger package distributions. Extensions are also much easier to unit test in isolation from the base object.
- Reducing upgrade costs through extension-based customizations reduces the support matrix for partners as fewer release combinations will need to be supported.

For these reasons, we have gradually been sealing the product models, so they only support extension-based customizations. AppPlatform and AppFoundation were the first. These models were sealed for overlayering in Platform update 3 (November 2016). Binary updates are now provided to these models on a monthly basis, achieving our goals of reducing upgrade cost and delivering innovation to our customers at a faster cadence.

With Microsoft Dynamics 365 for Finance and Operations release 8.0, we have sealed all product models. Now only extension based customizations are supported.

The following illustration shows the roadmap we followed as we moved to extensions, away from overlayering.
A soft seal results in a compiler warning upon overlayering. A hard seal results in a compiler error upon overlayering.

The Modern support policy provides three years of support for a release. Given this, overlayered code will continue to be supported for three years starting November 2017 on the Microsoft Dynamics 365 for Finance and Operations, Enterprise edition 7.3 release. However, this code will not be moved forward to subsequent product releases until the overlayered code is moved to extensions.

There is a substantial amount of work for Microsoft, partners, and customers to accomplish this goal. Workshops, office hours, Help topics, and additional resources are available for training and collaboration in this ecosystem. Internally, we are ready to build more extensibility features in both the core platform and the application. We’re working closely with partners with applications on AppSource to define patterns as they migrate to extensions.

The benefits of reducing upgrade friction and enabling innovation uptake will be worth the effort to remove overlayering.
Finance and Operations applications exclusively use extensions to customize the product. We're aware that this change impacts our entire partner ecosystem. We recommend that you read the resources listed on the Extensibility home page. These resources answer many questions and prepare you for building solutions using extensions.

You will discover that some customizations, which were possible with overlayering, cannot be done through extensions. To enable the same business requirements without overlayering, we have added many extension capabilities and expect to add more going forward. For some customizations that were done with overlayering, you will need to log requests, to make us aware of what you need.

What we are doing

We've been working toward an extension-based customization model for some time. Over the past several releases we have been gradually sealing models. As of Dynamics 365 for Finance and Operations release 8.0, this completes the sealing. From this release forward, only extension-based customizations are allowed.

In future releases, we will be adding even more extensibility capabilities to enable independent software vendors (ISVs) and value-added resellers (VARs) to deliver complete business solutions. We will prioritize these on a customer-by-customer basis with frequent releases.

How do I log extensibility requests?

If you discover a customization that you cannot implement as an extension, you must log a request to Microsoft to ensure appropriate extension support is added to the product for your scenario.

Before logging the request, there are a few things to consider:

- Could the requirement be met with existing extensibility features? Building solutions with extensions requires different design and implementation patterns.
- How important is the requirement to the customer and/or business analyst?
- Will the implementation be upgrade friendly for the long term?

Learn more about extensibility by reading the resources listed on the Extensibility home page and related resources.

Extensibility requests are logged using a specific project in LCS. Logged requests are collected under that same project. We recommend that you log related requests under the same LCS project as this helps maintain a holistic view on all requests for a specific solution or an implementation. Microsoft then further identifies logged requests by the organization name that is associated with the LCS account.

In your LCS project, at the top of the page, select the hamburger icon and then click Support menu item.
You can view the list of logged extensibility support requests and their status. Click the request ID to review details of the logged request.

New requests that are logged are briefly assigned a status of **Pending** while the request is copied to the Microsoft tracking databases. Next, an ID is assigned to the request and the status is updated to **Active**. When a request is processed by Microsoft, the status of the request will be updated to **Closed**. Click the request ID to view resolution date and description information. Requests that have been closed are released with monthly application updates.

**NOTE**

At this time, there is no state available to indicate feedback from Microsoft on when requests have been planned.

The **Extensibility support request** form includes two actions:

- Manage extensibility options
- Create extensibility support request

When you click **Manage extensibility options**, you can view all of the information that is shared between requests. This information includes, requests for either an **ISV** or **VAR** solution and if the requests are specific for a **Customer** implementation project. If the role selected **ISV** or **VAR**, a solution name must be specified. The name should be recognizable and correlate with AppSource solutions. The **Required by date** indicates the last date that requests can be made to be available for your development.

**IMPORTANT**

Note that Microsoft does not guarantee that all requests will be provided by the date given. However, the required date provides an indication that will be considered when planning for requests at Microsoft.
Extensibility support options can be updated to reflect any changes after a request is created. After you have made your updates to the request, click **Update** to notify Microsoft of your changes.

**NOTE**

There is currently no option with the tool to record what a Customer implementation includes regarding ISV or VAR solutions.

The action ‘Create extensibility support request’ is used for creating, or logging, new extensibility requests. When you log an extensibility request, provide detailed information about what you need to become enabled for extensibility, and include information on what it is you need to extend. This will help Microsoft to be efficient in addressing your requests. You are welcome to propose how Microsoft could enable the functionality that you need in the standard application in a way that effectively addresses your needs.

When you select the request type, determine how your request aligns with the request types that Microsoft uses to categorize requests. Each request type changes the form to include specific fields related to the request type. This helps guide the process to make the request actionable for Microsoft. Be sure to provide accurate names when naming elements and methods. Microsoft rarely enables requests by adding inline delegates, so when possible, consider other types of requests. Common application request types include **extract method**, **extensible enum**, **construct with throw**, and **method change**. Additionally, there is **platform request** and **metadata change** for proposing changes, including general platform improvements. The request type **method signature change** is typically for a breaking change. It is unlikely that a breaking change can be accommodated under a monthly update as it will require a more major release version to drive such changes.

Click **Attach file from computer** to upload documents that you can attach to requests. You can use the attachments to supply code snippets that provide additional details for the request. We recommend that you be as specific as possible with your requests.
Log a request for each instance. Do not bundle multiple requests into one. If multiple requests are related, consider adding a document or description that includes request ID's so that any work on the requests is considered in context.

The requests include a point of contact. This is needed for times when the logged extensibility request is not actionable for one reason of another. The requests may require discussion regarding, for example different design options. Microsoft will use this contact information to drive such interactions. Click Submit when you are ready to submit the request to Microsoft. Because requests can’t be edited after they are submitted, verify the data before you submit. Requests that are accidentally submitted with incomplete or inaccurate data can be removed using the designated action after clicking the ID on the request. Requests that are submitted to Microsoft will temporarily show as Pending until the request is created within the Microsoft tracking databases. This will assign an ID to the request and the state will become Active. This status update indicates that the request is now visible to Microsoft.

Make sure to read through the privacy statement before you log any requests.

<table>
<thead>
<tr>
<th>NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>We will not release extensibility requests as hotfixes.</td>
</tr>
</tbody>
</table>
Extensibility requests are exclusive for the application. We are not planning to accommodate extensibility requests for Dynamics AX 2012 or earlier releases.

**When will my extensibility requests be enabled?**

Extensibility requests are logged to a backlog. Microsoft engineers prioritize all requests, and then work on them in priority order. Please note that Microsoft is not ensuring that all requests will be fulfilled. In particular, requests that are intrusive by nature will not be supported, as they will prevent seamless upgrade.

**How will extensibility requests be made available to deploy?**

After Dynamics 365 for Finance and Operations release 8.0, we plan to release frequent application updates with new extensibility requests. This will follow the same release cadence as platform updates.

**Still have questions?**

Read the [Extensibility FAQ](#) and the other resources listed on the [Extensibility home page](#).
Will source code be available after the hard seal?
Yes, source code will be available after the hard seal. It's required for effective implementation and debugging.

How do I contact Microsoft if I have an extensibility request?
There is a special extensibility request form on the Lifecycle Services (LCS) site.

Where can I ask questions about extensibility patterns?
You can gain access to the Operations Extensibility group in Yammer. Operations Extensibility is an active group that has a significant amount of partner engagement. You get access via the Connect site by signing an NDA.

Where can I find documentation about extensibility patterns?
Documentation about extensibility patterns is available on the Extensibility home page.

Where can I get information about extensibility training?
We will announce training sessions in multiple ways. AppSource partners might receive direct invitations for some sessions. We will also announce workshops in the Operations Extensibility Yammer group and other forums.

What is the goal of sealing the application?
The application is being sealed as a step toward reducing upgrade costs in the ecosystem, so that customers can stay current on new releases. Customers can take advantage of new innovations that come from Microsoft and partners.

Extension packages enable better performance at design time, faster build automation, and unit testing. They also provide more efficient distribution and installation of models from independent software vendors (ISVs) and customers across different systems.

What is Microsoft working on to support this move?
There are several areas where the product team is working to improve the extensibility of the product. This work ranges from platform changes that have broad impact to refactored application code that provides additional hook points. For details, see the Operations Extensibility Yammer group and the product release plans.

After the application is sealed, what should customers do in a critical situation if they must make a quick change?
This scenario is very similar to a scenario where a critical bug fix is required, and the same process should be followed. As a required first step, you must create a case for support.

Can I overlayer an ISV solution after the hard seal of the application
We recommend that ISVs also seal their models. This step helps achieve the broader goal of reducing upgrade costs.

**Will I be able to overlayer an on-premises solution?**

On-premises solutions will follow the same patterns as cloud solutions. Therefore, no overlayering of Microsoft code will be supported.

**How often will Microsoft provide external updates so that partners can see what extensibility enhancements have been made?**

We plan to provide monthly updates of platform and application after Microsoft Dynamics 365 for Finance and Operations release 8.0.

**Why wasn't my extensibility request accepted?**

Some extensibility requests break changes. Some of the more common potentially breaking requests are listed here along with potential workarounds. In addition, read Creating extensions to understand the existing platform extension capabilities and Tips for logging extensibility requests to learn more about how to create solid requests if a capability doesn't exist in the latest release.

**Why can't EDT.StringSize be made extensible?**

- **Request:** Make EDT.StringSize changeable via extension.
- **Problem:** When a table string field (FieldX) is of type "parent EDT" and is associated (through table relations) with another table's field (FieldY) of type EDT2 (EDT2 is derived from "parent EDT"). If FieldY could have a larger string by allowing EDT2.StringSize to increase, FieldX would not be able to handle the new string size.
- **Workaround:** Create a new EDT and use that for the table field FieldY.

**Why can't a unique table index be made extensible?**

- **Request:** Make unique table indexes changeable via extension, for example by allowing an extra field to be added.
- **Problem:** If a unique table index changes and any data does not conform to the new index, then it would be a breaking change. Also, any query would affect it since it can now retrieve a non-unique record. For example, if a Person table had a key of "Name" and select person where name="Chris" works, but if BirthDate was added to the key, now there could be multiple records returned for "Chris".
- **Workaround:** Add "soft" constraints in the validateWrite or validateInsert methods.

**Why can't CountryRegionCode be made extensible?** *(it already is)*

- **Request:** Make CountryRegionCode changeable via extension.
- **Problem:** Starting with Platform update 14, changes to CountryRegionCode are supported if the CountryRegionCode property already has a value. Empty CountryRegionCode properties cannot be changed because that change is more restrictive (the element would now only be available for certain countries/regions) and therefore would be a breaking change.
- **Workaround:** Use the existing CountryRegionCode extension capability when the element is already country/region specific.

**Why can't the Table Field properties AllowEdit, AllowEditOnCreate, Mandatory, or IgnoreEDTRelation be made extensible?**

- **Request:** Make Table Field properties AllowEdit, AllowEditOnCreate, Mandatory, and/or IgnoreEDTRelation changeable via extension.
- **Problem:** The ability to change the "Allow Edit", "Allow Edit On Create", "Mandatory", and "IgnoreEDTRelation"
properties on Table Fields would result in breaking changes. Changing a field to allow editing changes the intent of the field. Not allowing a field to be edited can break existing behavior. Changing a relation breaks the original intent of that relation, which is a breaking change. Making a field mandatory can result in breaking existing behavior.

- Workaround: Add new Table Fields via extension and control those as needed.

**Why can't Security Privileges be made extensible?**

- Request: Make Security Privilege changeable via extension.
- Problem: The ability to change the Security Privilege would result in breaking changes because this is the lowest level of security metadata.
- Workaround: Create a new Security Privilege if needed and use that.

**Why should I avoid calling and extending APIs that are marked with InternalUseOnlyAttribute?**

Throughout the application, an effort has been made to avoid breaking changes to APIs made by customers, partners, or ISVs. When a class or method has the `InternalUseOnlyAttribute` applied to it, this means that the API is for internal use only and could change without warning. If customers, partners, or ISVs use or extend an API with `InternalUseOnlyAttribute`, this could create issues because the API could change at any time, which would require changes in their extensions before an update can be applied. This could result in urgent changes and the need to recompile. Developers should not depend on these classes and methods remaining unchanged.

Calls to classes and methods with the `InternalUseOnlyAttribute` will result in compiler warnings. Starting in Platform update 20 to Platform update 24, targeting classes and methods with `InternalUseOnlyAttribute` using Chain of Command will result in compiler errors. In Platform update 25 and later, we plan to continue to issue compiler warnings.
Introduction

When the application was first released, we strongly recommended that extensions be used instead of overlayering for customization. Overlayering-based customizations have been migrated from release to release through code migrations, and many customizations of application code are still based on the overlayering of code. For most partners, at least some of their solution is still based on overlaying, and some partners will have lots of overlayering across their solutions.

The amount of work that is required to change an implementation from overlayered code to extensions depends on the code itself. Some overlayered code can be changed relatively seamlessly. However, for some changes, you must rethink the customization to find an appropriate way to accomplish it through extension. Therefore, it can be a major undertaking to change complete solutions where multiple places have overlayered code. Such an undertaking requires an investment in the solution. The upside of this investment is a more seamless upgrade process, because customization is now based on application programming interfaces (APIs) through extensions. Additionally, a lengthy code upgrade process is no longer required as it was for overlayered code. More importantly, daily servicing of a running environment offers many benefits. The core application and extensions no longer have to be compiled together, and patching can be done by deploying precompiled assemblies. Therefore, customers can apply patches to their system in a relatively seamless manner, and the amount of downtime is minimized. However, there is work that must be done before this result can be achieved.

Although there are multiple ways to approach this task, we have gained experience through our close work with independent software vendors (ISVs) and value-added resellers (VARs) that have already started to migrate from overlayering to extensions. In this topic, we share some of this experience.

First things first

The task ahead is substantial, and we want to make sure that our shared investment pays dividends. Keep the goal in mind as you work through your customizations. When customization is done correctly, your solution has these qualities:

- It has no intrusive customizations.
- It supports side-by-side deployment with other ISV solutions.
- It's resilient to changes in Microsoft code.
- It's resilient to changes in other ISV solutions.
- It can be upgraded automatically to future versions.

This type of customization represents a fundamental shift of approach. Previously, the primary objective was to implement the functional requirements on the current version. This objective was acceptable, because we knew that manual work was required in order to upgrade the solution. Previously, great engineers minimized the manual upgrade cost. Now, every engineer must implement solutions that require zero effort to upgrade.

Staying on the right path

Cars are designed to be safe. However, they can't yet prevent accidents. Accident prevention remains the driver's responsibility. Similarly, the development toolset is designed for extensibility. However, the toolset can't yet prevent every type of intrusive customization. As an engineer, it's your responsibility to avoid intrusive customizations.

Sometimes, you might find that you can reach your functional goal only by implementing intrusive customizations. In this case, you should reach out to Microsoft to find a correct solution. You should not force
Obtain an overview of your code

When you create an overview of your code, first consider analyzing each of your solutions independently instead of analyzing them all together. This approach might be practical even if different teams work on the individual solutions. By choosing one team that you will engage before the other teams, you can gain some experience. Experience is valuable, because it not only helps you analyze and plan the work, but also helps the team ramp up and become familiar with the extensibility model. Therefore, the experience that you and your team gain can become valuable "lessons learned" that you can apply to later solutions.

We have gained practical experience both with ISVs that take each ISV solution in turn, and with ISVs that work as VARs and take customer solutions later.

No matter how the work is pieced together in solutions, you can use the Customization Analysis Report (CAR) to get information about what has been overlaid. This report is generated when you submit your solutions to the Code Migration tool on Microsoft Dynamics Lifecycle Services (LCS). The report is in Microsoft Excel format and includes a list of all the places that have overlaid code. You can use the report to both analyze and categorize all overlaid instances in your solution.

To obtain an overview, you might find it helpful to categorize each overlaid instance. The category that you apply to an overlaid instance should represent the approximate effort that is required in order to change the customization to extensions. Some customizations will be easily changed to extensions. However, for other customizations, the change will be more difficult.

From our experience working with numerous ISVs, we have found that the following categories are a good starting point.

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extensible enums</td>
<td>You can add new enum values by using extensions. For more information, see Add values to enums through extension.</td>
</tr>
<tr>
<td>Construct with throw</td>
<td>Most construct methods are simple and can be extended by using post-event handlers. However, some construct methods are more complex and throw an exception when no class is created.</td>
</tr>
<tr>
<td>Exposing members</td>
<td>Member variables that have the private access modifier in their definition can't be accessed through extensions unless they become exposed through public methods. You can request that we add access to members through extensions that currently have not been exposed for this. Note that access to protected members is generally enabled through extension classes.</td>
</tr>
<tr>
<td>Data manipulation methods that don't raise DataEvents</td>
<td>In some places in the application, data methods such as insert() and update() don't call super(). Therefore, the methods don't raise DataEvents to add extensions to. Microsoft plans to refactor the standard application so that it includes additional methods that enable extensions in these places. If you submit a request for Microsoft to add this, add any of the affected methods that you must currently overlay, if those methods haven't already been accounted for</td>
</tr>
<tr>
<td>CATEGORY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Extract method</td>
<td>This category is for code changes in the middle of methods, which can't be made through chain of command. When you request a method extraction, be sure to specify which lines of a method to extract, and what the signature of the new method must be.</td>
</tr>
<tr>
<td>SQL statement operations</td>
<td>SQL statements that are written directly in the application code don't enable extensions. When you make a request to extend these SQL statements, be sure to explicitly specify what you must extend, such as a field list, <code>where</code> clauses, or ordering.</td>
</tr>
<tr>
<td>Metadata overlayering</td>
<td>Provide the Application Object Tree (AOT) path of the element where you believe the metadata (property value) must be changed. Metadata changes can't be made through the current extension capabilities.</td>
</tr>
<tr>
<td>Method overlayering</td>
<td>This category is for customizations where a method is overlaid. You should consider converting the overlaid method to an extension, so that changes are clean by extension not substitution.</td>
</tr>
<tr>
<td>Method signature changes</td>
<td>The capability to change method signatures through overlayering will be discontinued. Other patterns for achieving similar results are required. You can request changes to the standard signature to support extensibility. Be sure to include information about additional parameters that are required.</td>
</tr>
<tr>
<td>Inventory dimensions</td>
<td>You can no longer add dimensions by editing the macro and recompiling the standard application. Another approach will be offered that involves predefined dimensions that are deployed at runtime. This approach drives changes to existing customizations where new dimensions are added.</td>
</tr>
<tr>
<td>Extensibility platform</td>
<td>Some customizations might not be possible through extensions unless new platform features are added. If you determine that customizations can't currently be done through extensions, open an extensibility request that explains the scenario and what is required.</td>
</tr>
<tr>
<td>Reports</td>
<td>Customizations of report designs have limited support for extensibility. In general, a new report must be created. Data provider classes can also be customized so that they include additional information. In some places, the standard application must be changed to enable this type of customization.</td>
</tr>
<tr>
<td>Other</td>
<td>This category is for overlayering instances that don't fit into any other category.</td>
</tr>
</tbody>
</table>

By categorizing all overlaying code, you gain an overview of what must be changed.

Analyzing for impact and estimating work

In a typical approach to assessing work impact, you break down tasks into something tangible. This approach also applies to this work. The categories that were discussed in the previous section help frame similar
customizations, and a first pass on estimates can be built by coming up with an overall estimate for each of these categories and similar categories that make up your solution. The group of customizations in a given category often has a few extremes that stand out, and it might be appropriate to establish estimates for these customizations individually.

Consider that some customizations will require either a request to Microsoft to enable extensibility or significant refactoring of the customization so that it can be done through extensibility. Both of these scenarios will increase the estimates for migrating the solution.

Customization that drives what are referred to as intrusive changes is often more complex to convert to extensions. For these changes, you must consider what is the correct way to approach the customization. Here are some examples of these changes:

- Customizations that request inline delegates.
- Customizations of complex classes or methods such as `SalesLinetype`.
- Changes to method signatures.
- Additions of inventory dimensions.
- Changes to report definitions and report data provider classes.
- Intrusive changes to forms.

For changes that require different approaches to make the customizations extension-based, you might have to log requests to Microsoft to enable extensibility. When creating your migration schedule, you'll need to take into account the delay of waiting for updates from Microsoft.

**What is supported, and what requires an extensibility request?**

When you review a customization, be sure to consider different options for converting it to an extension. Be sure to consider whether a method is hookable, or whether it can be a class extension or form event. Review most of the currently available application code that is available to you.

You might conclude that a change to the standard application is required in order to enable the required extension. In this case, you must log an extensibility request. The request is then put into the backlog at Microsoft so that it can be addressed. Don’t log extensibility requests by opening a request for a hotfix, because Microsoft doesn’t release extensibility requests as hotfixes.

Be sure to supply enough contextual information in your extensibility requests. For example, a request for an inline delegate might come from the current customization approach. However, to better accommodate extension, the requirement that led to this customization might be better served by a structural change to the standard application. We appreciate suggestions of this type, because they help move the application toward a better platform for building different customizations.

**Planning the migration**

Be sure to start planning the migration of your solutions early. This planning is important because it helps you make sure that you have room in your schedules to identify and log extensibility requests, and that you have room for the time delay before these requests become available in product releases. Additionally, acknowledge that your developers might have to build new skills, and make sure that you cater to any required learning as part of the migration plan.

Your solution might contain intrusive customizations that aren't easily accommodated through extensions. You should consider whether the business value of these customizations outweighs the effort of building them through extensions. In some cases, partners have decided to discontinue parts of their solutions, because they found that it was impractical to rebuild those parts through extensions, and those parts weren't critical to the solutions.
Some smaller fixes that you're customizing across the application might not be core for your solution, but they are important for the customers that you engage with. In these cases, you must decide whether you prefer to ask Microsoft to implement similar capabilities in the standard application. You can enter an extensibility request for this purpose. For example, if customers want to simplify standard business processes in the system, you might suggest that we add options for disabling steps of the process in the standard application.
In this tutorial, you’ll become familiar with the Fleet Management Extension model. This model contains elements that extend the functionality of the Fleet Management application. You can customize model elements by creating extensions. Unlike the overlaying capabilities of Microsoft Dynamics AX 2012, extensions don’t overlay the baseline model elements. Instead, extensions are compiled as a separate assembly that adds to or customizes the model and the associated business logic. You can extend metadata, for example, by adding a field to a table or adding a control to a form, and also extend or customize business logic by defining event handlers and plug-in classes. You can now author event handlers on several predefined events on tables, forms, form data sources, form controls, and others. Plug-ins are also a new extensibility concept that enables replacing or extending the business logic of the application.

Prerequisites

This tutorial requires you to access the environment using Remote Desktop, and that you are provisioned as an administrator on the instance.

Understanding the Fleet Management model

The Fleet Management application provides a rental car company a system for managing vehicles, customers, and vehicle reservations. The application is designed for use by the Fleet Clerk and Fleet Manager personas.

**Fleet Clerk**

The Clerk is the front desk employee who handles the face-to-face and over-the-phone interactions with customers. The Clerk is primarily concerned with entering customer information into the application, creating vehicle reservations for customers, upselling the reservation by offering vehicle accessories, and processing vehicle returns upon completion of a vehicle rental. The Clerk spends the vast majority of their time using the Fleet Management Workspace to prepare for interactions with customers by anticipating their needs and providing a pleasant and memorable experience, while interacting with the customer.

**Fleet Manager**

The Manager is the back office employee who handles setting business requirements and processes. The Manager is primarily concerned with entering vehicle information, defining the available vehicle accessories, vehicle maintenance, determining pricing, and analyzing business performance measures such as revenue, upsell success, and so on. The application’s business logic revolves around the following three primary entities and the relationships between them.

**Customers**

Customers contact the Fleet Clerk to make vehicle reservations, choose vehicle accessories, check out and return vehicles, and pay for vehicle rentals. Customer-related information is stored in the table named `FMCustomer`.

**Vehicles**

Vehicles vary primarily in their price, which is proportional to the vehicle class. The names of tables that store information about vehicles begin with `FMVehicle`.

**Reservations and rentals**

Reservations handle the relationship between customers and vehicles. Reservation information includes reservation dates, customer information, vehicle selection and price, and additional charges such as accessories or fees. Reservation and rental information is stored in the `FMRental` and `FMRentalCharge` tables. A
Extending the Fleet Management model

The basic Fleet Management application has been customized with additional capabilities that enable a rental car company to provide pricing incentives to its customers through discounts. The additional business logic and data that enables these discount capabilities is stored in the Fleet Management Extension model. The discount capabilities add value to the Fleet management application through three primary customizations.

The Fleet Management Extension data model

Two new tables have been added that store discount-related information. \texttt{FEDiscounts} stores the list of all discounts and their rates. \texttt{FERentalDiscountRelationTable} keeps track of the reservations that the discounts are applied to. Existing tables have been extended to account for the addition of discounts to the pricing scheme.

The table that keeps track of the vehicle rate for a particular reservation, named \texttt{FMVehicleRental}, has been extended to accommodate discounts to the vehicle rate. The table that keeps track of the accessories for a reservation, named \texttt{FMVehicleRentalCharge}, has been extended to accommodate discounts applied to accessories.

The Fleet Management Extension Calculation Engine

The basic calculation engine has been customized to add the various pricing schemes defined by the new discounts. A plug-in class has replaced the functionality of the base calculation engine. When a vehicle is reserved for more than 7 days, the vehicle Fleet Management model calculates savings based on the difference between a vehicle's daily rate and a lower weekly rate. The plug-in removes the weekly rate calculation because this same behavior can be accomplished by using discounts.

The Fleet Management User Interface Extensions

The Rental, which is contained by the form named \texttt{FMVehicleRental}, has been extended to enable the Clerk to apply discounts to a reservation. The on-screen price summary is updated in real time with savings information related to discounts that can be applied to vehicles and accessories related to the reservation. In the following steps, you'll explore the customizations that have been made in the Fleet management Extension model, as well as re-implement a portion of the customizations for yourself.

Setup

If you haven't opened the Fleet Management Solution in a previous tutorial, follow these steps. The fleet management solution file is available on the Dynamics AX downloadable VM.

1. On the \texttt{Desktop}, double-click the \texttt{Visual Studio} shortcut to open the development environment.
2. Open the \texttt{FleetManagement} solution. On the \texttt{File} menu, point to \texttt{Open}, and then select \texttt{Project/Solution}.
3. Browse to the desktop and open the \texttt{FleetManagement} folder. If the solution file is not on your computer, the steps to create it are listed in Tutorial: Create a Fleet Management solution file out of the Fleet Management models in the AOT.
4. Select the solution file named \texttt{FleetManagement}. The file type listed is Microsoft Visual Studio Solution.
5. Select \texttt{Open}. The solution may take some time to open.

Installing the demo data

If you've already installed the demo data, you can skip to the next section.

1. In the VM, open Internet Explorer and navigate to the application's base URL.
2. Sign in.
3. On the dashboard, open the navigation pane and navigate to \texttt{Fleet Management > Setup > Fleet Setup}.
4. Click Setup Demo Data.

![Fleet management configuration](image)

5. If you're prompted to reload the demo data, select Yes.

6. When the data is finished loading, select Close.

7. On the dashboard, open the navigation bar and navigate to System Administration > Common > Maintain aggregate measurements. (Steps 7 to 9 are not applicable on newer releases.)

8. Select FMAggregateMeasurements, and on the Action Pane, select Refresh now.

9. Wait until the processing completes. The ongoing processing is indicated at the top of the page by a series of moving dots. The processing is completed when the indicator disappears and the Time Last Processed field is updated.

**Open the FMRental form on the one-box environment**

1. In the VM, open Internet Explorer and navigate to the base URL of your Dynamics AX application. For more information, see Deploy and access development environments.

2. Sign in, if prompted.

3. Find the Reservation Management tile and select it to open the Reservation Management workspace.
4. When the Reservation Management workspace opens, select **Current rentals**.

5. The **Rental** form opens in grid view.

<table>
<thead>
<tr>
<th>RENTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VEHICLE RENTAL ID</th>
<th>VEHICLES</th>
<th>START DATE</th>
<th>END DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>00022</td>
<td>Adatum_Four_1</td>
<td>5/20/2015 09:40:12 PM</td>
<td>5/26/2015 09:40:12 PM</td>
</tr>
<tr>
<td>00025</td>
<td>AdventureWorks_Makalu_7</td>
<td>5/20/2015 09:40:12 PM</td>
<td>5/26/2015 09:40:12 PM</td>
</tr>
<tr>
<td>00028</td>
<td>Libware_LibwareFour_1</td>
<td>5/20/2015 09:40:12 PM</td>
<td>5/26/2015 09:40:12 PM</td>
</tr>
</tbody>
</table>

6. After the **Rental** form loads, select **Options > Change view > Header** to open the **Header view**.
7. When the Header view form loads, scroll to the bottom and expand the Discounts tab. This tab isn’t part of the Fleet Management model. It has been modeled in the Fleet Management Extension Model as an extension to the FMRental form.

8. Select Add to add a discount.

9. Select the Frequent Customer discount, and then select OK. The selected discount is added to the Discounts grid.

10. Use the shortcut key, Alt+F2, to open the FactBox.

11. Expand the Rental total FactBox on the right and view the discount savings that are applied.
Overview of the Fleet management discount extension project

In this tutorial, the FleetManagementDiscounts Project contains the model elements that belong to the model named Fleet Management Extension. Here, you'll explore and learn about the project elements.

Navigate to FM Renta l.Extension in the Tree Designer

1. In the Visual Studio, in Solution Explorer, in the FleetManagement Discounts project, expand User Interface > Form Extensions.

   ![Solution Explorer](image)

   The FM Renta l.Extension element is an extension element that extends the functionality of the FM Renta l. form by adding two new data sources and a new tab control.

2. In Solution Explorer, double-click FM Renta l.Extension to open the designer. As the following image shows:
   - The data sources shown in italic text are data sources defined in the baseline form.
   - The data sources shown in bold are the ones defined in the current extension.

   The designer presents an integrated view of the model element, including its extensions. Read-only nodes are shown in italic text, while nodes that belong to the current extension are shown in bold, with other visual cues that indicate the type of customization.

3. In the designer's search box, type 'e:' as shown in the image below. This filters the current designer to
only show nodes that belong to the current extension.

4. You can also type `e:LineViewDiscounts` to filter the designer to show nodes that match the name `LineViewDiscounts` and that belong to the current extension.

5. Expand the `LineViewDiscounts` node to see its contents.

Open the FM Rental Extension XML file to view the metadata

1. In the Solution Explorer, right-click FM Rental Extension form extension, and then click Open with.

2. In the Open with dialog box, select XML (Text) Editor, and then click OK.

3. When prompted to close the designer, click Yes.

4. Click the corresponding minus signs to collapse the child nodes of the Controls and Data Sources nodes. Refer to the following image for the correct result.
The XML file contains the metadata associated with the `FM Rental.Extension` element. You can see that this file contains metadata that describes only one tab page control and two data sources that are part of the extension. You can also see that it doesn’t contain any metadata from the base form.

View other elements in the Fleet Management discount extension project

The `Fleet Management Discounts` project contains two new tables, `FEDiscount` and `FERentalDiscountRelationTable`, and two extensions to existing Fleet Management tables, `FMRental` and `FMRentalCharge`.

1. In Solution Explorer, in Fleet Management Discounts, double-click `Data Model > Table Extensions > FMRental.Extension` to open the designer.

2. Expand the `Fields` node to see that this extension contains one added field, `FEVehicleRateDiscount`, to the base `FMRental` table.

3. Similarly, open the `FM RentalChange.Extension` element in the designer to explore its contents.

Inspect the data event handlers

In Solution Explorer, in the Fleet Management Discounts project, double-click `Code > Classes > FMRentalCharge_Extension` to open the code editor.
View the plug-in classes

In the event handler code of the FMRentalCharge_Extension class shown in the previous section, notice that both event handlers call FMTotalsEngineBase::GetInstance to retrieve the current instance of the Fleet Management calculation engine. The calculation engines are implemented by using plug-in classes. A class factory creates the appropriate instances of a plug-in class based on configuration or business data.

1. In the code editor window that displays FMRentalCharge_Extension.xpp, right-click GetInstance, and then select Go To Definition. The code editor opens with the abstract class FMTotalsEngineBase. This abstract class is called a plug-in point and it's associated with the following attribute:

Plug-in classes represent extensions or implementations of abstract classes or interfaces. Plug-in classes are associated with attributes defining their metadata and the plug-in point. In this example, there are two plug-in classes associated with the `FMTotalsEngineBase` plug-in point. The base calculation engine is defined by the plug-in class `FMTotalsEngine`. You can find it in the project `FleetManagement Migrated > Code > Classes`.

```csharp
class FMTotalsEngine extends FMTotalsEngineBase
```

The discount calculation engine is defined by the plug-in class `FEDiscountEngine`. You can find it in the project `FleetManagement Discounts > Code > Classes`.

```csharp
class FEDiscountEngine extends FMTotalsEngine
```

2. Look at the `GetInstance` method. It uses the plug-in factory `SysPluginFactory::Instance` to instantiate the current calculation engine based on current plug-in metadata. The plug-in metadata is specified in the global configuration table, `FMParameters`.

```csharp
    // the [key, value] pair from the SettingsJSONValue still should match the [key, value] pair from the attribute on the plugin class meta. SetManagedValue("TotalEngine", FMParameters.TotalEngine);
    _instance = SysPluginFactory::Instance("Dynamics.AX.Application.FMTotalsEngineBase", classStr(FMTotalsEngineBase), meta) as FMTotalsEngineBase;
    if (!_instance)
    {
        warning("SysPluginFactory could not initialize totals engine - using default. Consider running Fleet setup.");
        _instance = FMTotalsEngine::construct();
    }
```

The Finance and Operations apps also support configurable plug-in classes where the plug-in metadata associate with the class isn’t known at development time and is configurable at runtime by an administrator. This tutorial doesn’t cover that feature.

Create additional Fleet Management extensions

This section shows how you can use the Visual Studio tools to create and interact with extensions.

**Extend the FMVehicle Table**

1. In **Solution Explorer**, select the **FleetManagement Discounts** project.

2. In **Visual studio**, in **Application Explorer**, select **View > Application Explorer**, and search for the table named FMVehicle. Type `FMVehicle type:Table` in the filter bar and press **Enter**.

```csharp
 FMVehicle type:Table
```

---

**Create additional Fleet Management extensions**

This section shows how you can use the Visual Studio tools to create and interact with extensions.

**Extend the FMVehicle Table**

1. In **Solution Explorer**, select the **FleetManagement Discounts** project.

2. In **Visual studio**, in **Application Explorer**, select **View > Application Explorer**, and search for the table named FMVehicle. Type `FMVehicle type:Table` in the filter bar and press **Enter**.
3. Right-click FMVehicle, and then select Create extension.

![Image of application explorer]

An extension of the FMVehicle table is created in the FleetManagement Discounts project named FMVehicle.Extension.

4. In Solution Explorer, right-click FMVehicle.Extension, and then select Open with. In the dialog box, select XML (Text) Editor, and then select OK. Note: This extension file is simply a template that doesn't contain metadata from the base FMVehicle table. An extension file will always contain only the metadata that defines the extension and nothing from the base model element.

```xml
<?xml version="1.0" encoding="utf-8"?>
<xAxTableExtension xmlns="http://www.w3.org/2001/XMLSchema-instance">
  <Name>FMVehicle.Extension</Name>
  <FieldGroupExtensions />
  <FieldGroups />
  <Fields />
  <Indexes />
  <Relations />
</xAxTableExtension>
```

5. Close the XML editor.

6. In Solution Explorer, double-click FMVehicle.Extension to open the designer.

7. Right-click Fields and add a new integer field. Change the name of the field to NumberOfCylinders.

8. In the Properties window, set the Label property of the new field to NumberOfCylinders.

9. Drag-and-drop the NumberOfCylinders field into the AutoReport field group to extend the field.

11. Expand the **Events** node. The **Events** node lists all events that the table exposes. This list includes events that are defined by the framework, and delegate methods that are defined by application developers.
NOTE
Different framework events are exposed on the designers of many types of element and sub-elements, like table events, form events, form data source events, and form control events.

12. Right-click onValidatedWrite, and then select **Copy event handler method**.

This step copies the event handler method signature to the clipboard.

13. Add a new class named **FMVehicleEventHandlers** to the **FleetManagement Discounts** project.

14. In **Solution Explorer**, double-click **FEVehicleEventHandlers** to open the code editor.

15. Right-click and paste the event handler method that you copied in step 12.

```csharp
Class FMVehicleEventHandlers
{
    /// <summary>
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    [DataEventHandler(tableStr(FMVehicle), DataEventType::ValidatedWrite)]
    public static void FMVehicle_onValidatedWrite(Common sender, DataEventArgs e)
    {
    }
}
```

16. Insert the following code into the **FMVehicle_onValidatedWrite** event handler. This code validates that the number of cylinders can't be greater than 8.

```csharp
[DataEventHandler(tableStr(FMVehicle), DataEventType::ValidatedWrite)]
public static void FMVehicle_onValidatedWrite(Common sender, DataEventArgs e)
{
    ValidateEventArgs validateArgs = e as ValidateEventArgs;
    FMVehicle vehicle = sender as FMVehicle;
    boolean result = validateArgs.parmValidateResult();
    if (vehicle.NumberOfCylinders > 8)
    {
        result = checkFailed("Invalid number of cylinders.");
        validateArgs.parmValidateResult(result);
    }
}
```

17. Save FMVehicleEventHandlers class
Extend the FMVehicle Form

Next, add an extension to the FMVehicle form in the FleetManagement Discounts project. First, be sure to select this project in Solution Explorer.

1. Use Application Explorer to find the form named FMVehicle, and in the Application Explorer filter bar, enter `FMVehicle type:form`.

2. Right-click the form, and then click Create extension.

3. Add a new integer control named NumberOfCylinders to the Attributes2 group control as shown below. You can find this control by expanding Design > Tab > TabPageDetails > TabHeader > DetailsDetails > Attributes2.

4. Bind the new control to theNumberOfCylinders data field in the properties window as follows.

5. Save FMVehicle.Extension and build the project.

Test your extensions

1. In Solution Explorer, right-click FleetManagement Discounts, and then click Set as StartUp project.
2. Similarly, in FleetManagement Discounts, set the FMVehicle.Extension form as the startup object.

3. Press Ctrl+F5 to start without debugging, or use the Debug menu.

4. After the Vehicles form opens, select a vehicle to view its details.

5. Expand the Details tab and notice the new Number of Cylinders field.

6. In the Action Pane, click Edit, and change the value in the Number of cylinders field to 12.

7. In the Action Pane, click Save.

8. Notice the validation error.

9. Enter a valid number of cylinders, less than 9, and then save the new value.

**Experiment with event handlers on form controls**

You can add event handler methods on existing controls.

1. Find the AddLine command button control in the FMRental form designer, right-click the OnClicked event, and select Copy event handler method.

2. Paste the event handler method in a class of the Fleet Management Extension model and add X++ code to implement it.
When implementing the AddLine_OnClicked event handler, you can access the button control instance using the `sender` parameter.

```csharp
FormButtonControl button = sender as FormButtonControl;
```

If you need to access the parent form or any of its variables, this example shows how to access the `FormRun` instance and one of its data sources.

```csharp
FormRun fr;
fr = sender.formRun();
var frDs = fr.dataSource("FMRental");
```

### Experiment with event handlers on form data sources

Just like tables, form controls and other element types, form data sources and form data source fields provide framework-level events. The following example shows how you can use the `ValidatingWrite` event on a form data source or the `Validating` event on a form data source field to validate user input on the FMRental form. This functionality is available as of Platform Update 7.

```csharp
/// <summary>
/// When saving a new rental, prevent setting the start mileage on the FMRental form to a value that is equal to 1
/// </summary>
[FormDataSourceEventHandler(formDataSourceStr(FMRental, FMRental), FormDataSourceEventType::ValidatingWrite)]
public static void FMRental_OnValidatingWrite(FormDataSource sender, FormDataSourceEventArgs e)
{
    var datasource = sender as FormDataSource;
    var args = e as FormDataSourceCancelEventArgs;
    if (args != null && datasource != null)
    {
        FMRental record = datasource.cursor() as FMRental;
        if (record.recId == 0)
        {
            if (record.startmileage == 1)
            {
                boolean doCancel = !checkFailed("Start Mileage = 1 is not allowed");
                args.cancel(doCancel);
            }
        }
    }
}
```
/// <summary>
/// Prevent changing the start mileage field on the FMRental form to a value that is equal to 1
/// </summary>
[FormDataFieldEventHandler(formDataFieldStr(FMRental, FMRental, StartMileage),
FormDataFieldEventType::Validating)]
public static void StartMileage_OnValidating(FormDataObject sender, FormDataFieldEventArgs e)
{
    var dataObject = sender as FormDataObject;
    var args = e as FormDataFieldCancelEventArgs;
    if (args != null && dataObject != null)
    {
        var datasource = dataObject.datasource() as FormDataSource;
        if (datasource != null)
        {
            FMRental record = datasource.cursor() as FMRental;
            if (record.RecId > 0)
            {
                if (record.StartMileage == 1)
                {
                    boolean doCancel = !checkFailed("Start Mileage = 1 is not allowed");
                    args.cancel(doCancel);
                }
            }
        }
    }
}

Experiment with table extension display and edit methods

Extension methods enable you to extend tables by creating new display and edit methods on these tables without over-layering X++ code (Extension method must belong to a class named with an _Extension suffix). For example, this class shows how you can extend the FMVehicle table with an extension display method named CupHoldersDisplay.

```csharp
public static class FMVehicle_Extension
{
    public static display int CupHoldersDisplay(FMVehicle vehicle)
    {
        return 7;
    }
}
```

On a form or form extension, you can bind a control to this display method by setting "Data Source = FMVehicle" and "Data method = "FMVehicle_Extension::CupHoldersDisplay" as the image below shows.

<table>
<thead>
<tr>
<th>Data</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow Negative</td>
<td>Auto</td>
</tr>
<tr>
<td>Array Index</td>
<td>0</td>
</tr>
<tr>
<td>Auto Declaration</td>
<td>No</td>
</tr>
<tr>
<td>Cache Data Method</td>
<td>Auto</td>
</tr>
<tr>
<td>Configuration Key</td>
<td></td>
</tr>
<tr>
<td>Country Region Codes</td>
<td></td>
</tr>
<tr>
<td>Country Region Context Field</td>
<td></td>
</tr>
<tr>
<td>Custom Display Name</td>
<td>DisplayMethod (Integer)</td>
</tr>
<tr>
<td>Data Field</td>
<td></td>
</tr>
<tr>
<td>Data Method</td>
<td>FMVehicle_Extension::CupHoldersDisplay</td>
</tr>
<tr>
<td>Data Source</td>
<td>FMVehicle</td>
</tr>
</tbody>
</table>

Create a Fleet extension package for deployment

To deploy your extension to another environment, for example, a test, pre-production or production
environment, you must create a deployment package.

1. In Visual Studio, on the Dynamics AX menu, point to Deploy, and then select Create Deployment Package.

2. Select the Fleet Management Extension check box.

3. In the Package file location text box, enter "c:\FMLab".

4. Select Create. A deployment package that contains the Fleet management Extension package is created.

Additional resources

Download FMLab sample code
This topic discusses the two methods of customizing source code and metadata of model elements - overlayering and extensions and details supported extension capabilities.

Overlaying

You can customize source code and metadata of model elements that are shipped by Microsoft or third-party Microsoft partners. In order to customize metadata and source code of a model, the developer must create a new model that overlays the model they want to customize. For example, solution developers can provide code in the SLN layer, independent software vendors can use the ISV layer, and value-added resellers can use the VAR layer. Functionality defined in higher layers (VAR layer in this example) can override the functionality of lower layers. The overlaying model must belong to the same Package as the source model and belong to a layer that is higher than the source model. Overlayering is a powerful tool to perform advanced customizations of metadata and source code, but may increase the cost of upgrading a solution to a new version.

Extensions

You can customize an application by using extensions. An extension enables you to add functionality to existing model elements and source code. Extensions provide the following capabilities:

- Creating new model elements.
- Extending existing model elements.
- Extending source code using class extensions.
- Customizing business logic. Ways to customize business logic include:
  - Creating event handlers to respond to framework events, such as data events.
  - Creating event handlers to respond to event delegates that are defined by the application.
  - Creating new plug-ins.

To get started, review or complete this tutorial: Customize model elements through extension.

Extension models and packages

You can create a model that contains only new model elements, new code, or extensions. This model is compiled into its own separate assembly. These assemblies, along with related metadata and runtime artifacts can be packaged (as a deployable package file) and deployed on runtime sandbox or production environment. To create an extension model, go through the Create model wizard and select Create new package on the second step.
Extension models have several advantages, including:

- **Application lifecycle management (ALM):** Extension models simplify and improve the performance of deployments, builds, test automation and delivery to customers.

- **Design time performance:** Building your model or project doesn’t require you to recompile the entire application.

- **Servicing:** In the cloud, Microsoft can install, patch, upgrade, and change internal APIs without affecting your customizations.

- **Upgrades:** Unlike overlayering, extensions reduce the cost of upgrading to a new version, as this approach eliminates costly code and metadata conflicts.

The following diagram illustrates how extensions get isolated in their assemblies.
You can extend source code in 3 ways:

- By subscribing to events (framework events and delegates)
- By writing plug-ins.
- By creating class extensions (aka class Augmentation), see section below.

You should understand the following characteristics of framework events:

- Events are implemented as multi-cast delegates, which means that more than one event handler can be subscribed to any particular event.
- Events are broadcast; there's no sequencing of calls to event handlers.
- Event handlers execute within the transaction scope of the base methods.

**Events**

Events are raised as preceding and succeeding operations around the base methods. This means that you have the opportunity to run code before a base method is called and after it has completed. Microsoft Dynamics AX 2012 introduced XPP events, which are also available in this release and can be subscribed to in your extensions.

**Plug-ins**

Plug-ins are extension points that are defined by the base application. By using a class-factory pattern, plug-ins enable you to replace the base functionality. You can see how to implement a plug-in in the tutorial, Customize model elements through extension.

**Class Extensions**

Class extensions enable you to augment a class by adding methods and variables to existing classes, tables and forms. For more details, refer to the topic Class extension model in X++.

**Form extensions**
You can extend the functionality of a form by extending its controls and data sources. For example, in a form extension, you can:

- Add a new control.
- Enable or disable a control.
- Change the text or label property of a control.
- Change a control’s visibility.
- Change a form’s help text.
- Change a form’s caption.
- Add a new data source.
- Add a form part.

Other ways to customize a form, such as reordering controls in the form are planned to be included in a future release. In Microsoft Dynamics AX 2012, you could override form methods. In the current version, you use extensions to implement event handlers that are called from the base implementations of form methods. The following table lists each method and its associated events.

<table>
<thead>
<tr>
<th>Published Form DataSource Method</th>
<th>Preceding Event</th>
<th>Succeeding Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>active</td>
<td>N/A</td>
<td>Activated</td>
</tr>
<tr>
<td>delete</td>
<td>Deleting</td>
<td>Deleted</td>
</tr>
<tr>
<td>validateWrite</td>
<td>ValidatingWriting</td>
<td>ValidatedWrite</td>
</tr>
<tr>
<td>write</td>
<td>Writing</td>
<td>Written</td>
</tr>
<tr>
<td>create</td>
<td>Creating</td>
<td>Created</td>
</tr>
<tr>
<td>executeQuery</td>
<td>N/A</td>
<td>QueryExecuted</td>
</tr>
<tr>
<td>linkActive</td>
<td>N/A</td>
<td>PostLinkActive</td>
</tr>
<tr>
<td>init</td>
<td>N/A</td>
<td>Initialized</td>
</tr>
<tr>
<td>validateDelete</td>
<td>ValidatingDelete</td>
<td>ValidatedDelete</td>
</tr>
<tr>
<td>reread</td>
<td>N/A</td>
<td>Reread</td>
</tr>
<tr>
<td>selectionChanged</td>
<td>N/A</td>
<td>SelectionChanged</td>
</tr>
<tr>
<td>markChanged</td>
<td>N/A</td>
<td>MarkChanged</td>
</tr>
<tr>
<td>leaveRecord</td>
<td>LeavingRecord</td>
<td>LeftRecord</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Published Form Object Method</th>
<th>Preceding Event</th>
<th>Succeeding Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td>Initializing</td>
<td>Initialized</td>
</tr>
<tr>
<td>close</td>
<td>Closing</td>
<td>N/A</td>
</tr>
<tr>
<td>Published Form Object Method</td>
<td>Preceding Event</td>
<td>Succeeding Event</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>run</td>
<td>N/A</td>
<td>PostRun</td>
</tr>
<tr>
<td>activate</td>
<td>N/A</td>
<td>Activated</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Published Form Control Method</th>
<th>Preceding Event</th>
<th>Succeeding Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>modified</td>
<td>N/A</td>
<td>Modified</td>
</tr>
<tr>
<td>validate</td>
<td>Validating</td>
<td>Validated</td>
</tr>
<tr>
<td>leave</td>
<td>Leaving</td>
<td>LostFocus</td>
</tr>
<tr>
<td>enter</td>
<td>N/A</td>
<td>Enter</td>
</tr>
<tr>
<td>gotFocus</td>
<td>N/A</td>
<td>GotFocus</td>
</tr>
<tr>
<td>clicked</td>
<td>N/A</td>
<td>Clicked</td>
</tr>
<tr>
<td>selectionChange</td>
<td>SelectionChanging</td>
<td>N/A</td>
</tr>
<tr>
<td>pageActivated</td>
<td>N/A</td>
<td>PageActivated</td>
</tr>
<tr>
<td>allowPageDeactivate</td>
<td>AllowPageDeactivate</td>
<td>N/A</td>
</tr>
<tr>
<td>expand</td>
<td>Expanding</td>
<td>Expanded</td>
</tr>
<tr>
<td>tabChanged</td>
<td>N/A</td>
<td>TabChanged</td>
</tr>
<tr>
<td>dialogClosed</td>
<td>N/A</td>
<td>DialogClosed</td>
</tr>
</tbody>
</table>

**Code behind extension forms**

You can use class extensions to author X++ logic associated with form extensions. This allows the definition of state variables accessible to form and control event handlers. It also allows overriding form methods without overlayering code. Refer to this blog article for an example.

**Table extensions**

You can create a table extension to extend a table's design and logic. You can add new fields, field groups, indexes, mappings and relations. You can also add new fields to existing field groups, change the label of a table field, change the Created By, Created Date Time, Modified By, Modified Date Time properties. Using table extensions, you can also change the Extended Data Type property on fields and set it to an EDT that is derived from the current EDT (This is available as of platform update 8).

In Microsoft Dynamics AX 2012, you could override the virtual methods of a table's base class to control the behavior that occurred during table operations, such as when creating, reading, updating, or deleting. In the current version, you instead use extensions to implement event handlers that are called from the base implementations of the table methods. The following table lists each table method and its events.
<table>
<thead>
<tr>
<th>Published Table Method</th>
<th>Preceding Event</th>
<th>Succeeding Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>validateWrite</td>
<td>ValidatingWrite</td>
<td>ValidatedWrite</td>
</tr>
<tr>
<td>validateDelete</td>
<td>ValidatingDelete</td>
<td>ValidatedDelete</td>
</tr>
<tr>
<td>validateField</td>
<td>ValidatingField</td>
<td>ValidatedField</td>
</tr>
<tr>
<td>validateFieldValue</td>
<td>ValidatingFieldValue</td>
<td>ValidatedFieldValue</td>
</tr>
<tr>
<td>modifiedField</td>
<td>ModifyingField</td>
<td>ModifiedField</td>
</tr>
<tr>
<td>modifiedFieldValue</td>
<td>ModifyingFieldValue</td>
<td>ModifiedFieldValue</td>
</tr>
<tr>
<td>Insert</td>
<td>Inserting</td>
<td>Inserted</td>
</tr>
<tr>
<td>Update</td>
<td>Updating</td>
<td>Updated</td>
</tr>
<tr>
<td>Delete</td>
<td>Deleting</td>
<td>Deleted</td>
</tr>
<tr>
<td>Initvalue</td>
<td>InitializingRecord</td>
<td>InitializedRecord</td>
</tr>
<tr>
<td>FinalDeleteValidation</td>
<td>Executed when a delete operation is performed on a table object, before the operation is committed to the underlying database table</td>
<td>N/A</td>
</tr>
<tr>
<td>FinalInsertValidation</td>
<td>Executed when an insert operation is performed on a table object, before the operation is committed to the underlying database table</td>
<td>N/A</td>
</tr>
<tr>
<td>FinalReadValidation</td>
<td>Executed when a read operation is performed on a table object.</td>
<td>N/A</td>
</tr>
<tr>
<td>FinalUpdateValidation</td>
<td>Executed when an update operation is performed on a table object, before the operation is committed to the underlying database table.</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Validation events capture and return results by using the `DataEventArgs` parameter. The display and edit method modifiers are supported on table extensions.

**View and Data entity extensions**

You can extend a View or Data entity to achieve much of the functionality available with table extensions.

**Enum extensions**

You can extend any Enum that is marked extensible (`IsExtensible=True`).
By extending an Enum, you can add new Enum values to it. It is important to keep the following in mind when dealing with extensible Enums:

1. You cannot have X++ logic that depends on the integer value of Enum values (For example, \textit{If (Enum1.v1 > Enum1.v2)} ... is not supported for extensible enums)
2. When Enum values of extensible Enums are synchronized into the database:
   - Integer values that belong to the baseline enum are deterministic, they come from the metadata.
   - Integer values that are an extension are generated during the synchronization process and are not deterministic.

**EDT extensions**

You can extend an EDT element in order to modify any of the following properties:

- Form help
- Label
- String size
- Help text

**Query extensions**

You can extend a Query element to achieve the following:

- Add ranges to an existing data source.
- Add new (embedded) data sources to an existing data source.
- Add new fields to an existing data source.

**Menu extensions**

You can extend a Menu element to achieve the following:

1. Add new menu items, submenus, menu references and tile references to an existing menu.
2. Hide an existing menu item, tile, or sub-menu in a menu by setting the \textbf{Visible} property to No.

**Security role and duty extensions**

You can extend a Security Role or a Security Duty to add new duties/privileges to these elements.

**Report extensions**
You can customize reports and business docs using extensions, below is a list of tutorials that help you learn more.

Customize App Suite reports by using extensions: Customizations to reporting solutions in the standard application are fully supported using a pure ‘Extension’ model. This article offers guidance on how to add the most common customizations to standard application reports without over-layering Application Suite artifacts. Here are some...

Create custom designs for business documents: This article focuses on the steps involved in crafting a custom report design for an existing application business document using a ‘pure’ extension model. Follow the steps below to associate a custom report design with an application document instance....

Expand Application Suite report data sets: This article focuses on the expansion of an existing report data set produced using X++ business logic in a Report Data Provider (RDP) class. Use custom delegate handlers and table extensions to include additional field data and/or calculations without...

Extend report menu items to redirect user navigation: This article focuses on the process of extending existing application menu items to redirect navigations with minimal code changes. Using this technique you will avoid the hassle of tracking down and replacing all references to an existing application...

Label extensions

You can create label extension files in order to modify the string value of a label, add new labels to the same label file or add new languages. To create a label extension file you must name it with a _extension suffix. For example, to extend the FLM labels of the Fleet Management model, do the following:

1. Create a project that belongs to a model that references Fleet Management (The model Fleet Management Extension is an example).
2. Add a new label file to the project and name it FLM_Extension.
3. Within the FLM_Extension label file, you can create new labels or modify the value of labels that are defined in the FLM label file of the Fleet Management model. Use the standard label editor to define new labels or redefine labels that already exist in the original FLM label file.
4. If your goal is to create translations of the FLM label, right-click on the FLM_Extension element in your project and select Add new languages. Follow the wizard to add translation files to the FLM labels.

NOTE

If the FLM_Extension file already exists in another model, you can name your file FLM_ExtensionN where N is any integer (For example FLM_Extension2, FLM_Extension3, ...etc)

Extension of Country/Region Codes

NOTE

This functionality is available as of Platform update 7.

The Country Region Codes property enables developers to restrict functionality to certain regions or countries based on the current legal entity’s primary address. Developers can extend this functionality by setting the Country Region Codes property on the following extension element types: Menu extension, Menu Item extension, Table extension (and fields), Form extensions (form controls), EDT extensions, Enum extensions, and View extensions.

You can specify additional country/region codes in their extension. The effective country/regions (at runtime) associated with an element will be the union of all codes from the baseline element and all its extensions.
Event argument types

When an event takes place, the delegates described in the sections above get triggered. In this section, we provide the details of the types of the arguments that are passed as the event arguments. Some of the entries in the table below have a null in the column designating the event args; this means that no arguments are passed - the relevant information is in the first argument (typically called sender) in this case.

<table>
<thead>
<tr>
<th>EVENT</th>
<th>ARGUMENT TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>onDefaultedField</td>
<td>DefaultFieldEventArgs</td>
</tr>
<tr>
<td>onDefaultedRow</td>
<td>null</td>
</tr>
<tr>
<td>onDefaultingField</td>
<td>DefaultFieldEventArgs</td>
</tr>
<tr>
<td>onDefaultingRow</td>
<td>null</td>
</tr>
<tr>
<td>onDeleted</td>
<td>null</td>
</tr>
<tr>
<td>onDeletedEntityDataSource</td>
<td>DataEntityContextResultEventArgs</td>
</tr>
<tr>
<td>onDeleting</td>
<td>null</td>
</tr>
<tr>
<td>onDeletingEntityDataSource</td>
<td>DataEntityContextResultEventArgs</td>
</tr>
<tr>
<td>onFindingEntityDataSource</td>
<td>DataValidationEventArgs</td>
</tr>
<tr>
<td>onFoundEntityDataSource</td>
<td>DataEntityContextRecordEventArgs</td>
</tr>
<tr>
<td>onGettingDefaultingDependencies</td>
<td>DefaultingDependenciesEventArgs</td>
</tr>
<tr>
<td>onGotDefaultingDependencies</td>
<td>DefaultingDependenciesEventArgs</td>
</tr>
<tr>
<td>onInitializedEntityDataSource</td>
<td>DataEntityContextEventArgs</td>
</tr>
<tr>
<td>onInitializedRecord</td>
<td>null</td>
</tr>
<tr>
<td>onInitializingEntityDataSource</td>
<td>DataEntityContextEventArgs</td>
</tr>
<tr>
<td>onInitializingRecord</td>
<td>null</td>
</tr>
<tr>
<td>onInserted</td>
<td>null</td>
</tr>
<tr>
<td>onInsertedEntityDataSource</td>
<td>DataEntityContextResultEventArgs</td>
</tr>
<tr>
<td>onInserting</td>
<td>null</td>
</tr>
<tr>
<td>onInsertingEntityDataSource</td>
<td>DataEntityContextResultEventArgs</td>
</tr>
<tr>
<td>onMappedDatasourceToEntity</td>
<td>DataEntityContextEventArgs</td>
</tr>
<tr>
<td>onMappedEntityToDataSource</td>
<td>DataEntityContextEventArgs</td>
</tr>
<tr>
<td>EVENT</td>
<td>ARGUMENT TYPE</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>onMappingDatasourceToEntity</td>
<td>DataEntityContextEventArgs</td>
</tr>
<tr>
<td>onMappingEntityToDataSource</td>
<td>DataEntityContextEventArgs</td>
</tr>
<tr>
<td>onModifiedField</td>
<td>ModifyFieldEventArgs</td>
</tr>
<tr>
<td>onModifiedFieldValue</td>
<td>ModifyFieldEventArgs</td>
</tr>
<tr>
<td>onModifyingField</td>
<td>ModifyFieldEventArgs</td>
</tr>
<tr>
<td>onModifyingFieldValue</td>
<td>ModifyFieldEventArgs</td>
</tr>
<tr>
<td>onPersistedEntity</td>
<td>DataEntityContextEventArgs</td>
</tr>
<tr>
<td>onPersistingEntity</td>
<td>DataEntityContextEventArgs</td>
</tr>
<tr>
<td>onPostedLoad</td>
<td>null</td>
</tr>
<tr>
<td>onPostingLoad</td>
<td>null</td>
</tr>
<tr>
<td>onUpdated</td>
<td>null</td>
</tr>
<tr>
<td>onUpdatedEntityDataSource</td>
<td>DataEntityContextResultEventArgs</td>
</tr>
<tr>
<td>onUpdating</td>
<td>null</td>
</tr>
<tr>
<td>onUpdatingEntityDataSource</td>
<td>DataEntityContextResultEventArgs</td>
</tr>
<tr>
<td>onValidatedDelete</td>
<td>ValidateEventArgs</td>
</tr>
<tr>
<td>onValidatedField</td>
<td>ValidateFieldEventArgs</td>
</tr>
<tr>
<td>onValidatedFieldValue</td>
<td>ValidateFieldEventArgs</td>
</tr>
<tr>
<td>onValidatedWrite</td>
<td>ValidateEventArgs</td>
</tr>
<tr>
<td>onValidatingDelete</td>
<td>ValidateEventArgs</td>
</tr>
<tr>
<td>onValidatingField</td>
<td>ValidateFieldEventArgs</td>
</tr>
<tr>
<td>onValidatingFieldValue</td>
<td>ValidateFieldEventArgs</td>
</tr>
<tr>
<td>onValidatingWrite</td>
<td>ValidateEventArgs</td>
</tr>
</tbody>
</table>

**Development tools support**

The development tools in Visual Studio provide integrated features to help you create and work with extensions. For example, when you right-click an element name in Application Explorer, you can create an extension for
To create an extension, the current project in **Solution Explorer** must belong to a model that references the model of the selected element in **Application Explorer**. To view the model for a particular project, view the project properties.

Visual Studio creates the extension file for you, either in the current project or in a new project. You can then work with the extension file either as source code or by using a designer. You package a code-extension model for deployment exactly like you would package any other model. On the **Dynamics 365** menu, point to **Deploy**, click **Create Deployment Package**, and then select the check box for the package name.

**Framework events**

Tables, form data sources, form controls, and other element types that support extension events list the available events (and delegates) under an **Events** collection node. For example, viewing the **Events** node of a table extension shows events that are defined by the framework, and delegate methods that are defined by application developers.
Note: Events are exposed on the designer on different element and sub-element types, like table events, form events, form data source events, form control events, and others. Open the context menu of an event node to interact with events:

- **Copy event handler method**: This option copies a method signature to the clipboard. You can paste it in any X++ code editor to define a method that subscribes to the selected event.
- **Find event handlers**: Searches and lists all methods subscribed to the selected event.

Additional resources

*Customize model elements through extension*
This topic provides links to extensibility updates.

- Extensibility changes version 10.0.3
- Extensibility changes version 10.0.2
- Extensibility changes version 10.0.1
- Extensibility changes version 10.0
- Extensibility changes version 8.1.3
- Extensibility changes version 8.1.2
- Extensibility changes version 8.1.1
- Extensibility changes version 8.1
- Extensibility changes version 8.0.4
- Extensibility changes version 8.0.3
- Extensibility changes version 8.0.2
- Extensibility changes version 8.0.1
- Extensibility changes in version 8.0
- Extensibility changes in version 7.3
- Extensibility changes July 2017
This topic lists the extensibility features that were implemented in Microsoft Dynamics 365 for Finance and Operations version 10.0.3. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Enumerations made extensible

The following enumerations have been made extensible in this update:

- LedgerJournalWFApprovalModule
- RetailReceiptTransaction

SQL operations made extensible

The following SQL operations have been made extensible in this update:

- CustVendTrans support for an extensible map pattern

Metadata changes

The following metadata changes have been made in this update:

- CostSheetAmount.NoOfDecimalsIsExtensible
- SalesLinePercent.NoOfDecimalsIsExtensible

Refactored methods

The following methods have been refactored to support extensibility:

- BankReconciliationDataInitializer.initDocumentOpenTmp
- BankReconciliationDataInitializer.initStatementOpenTmp
- Class\BomCalcJob_All.processSingleTask
- Class\KanbanEventQuantityMap.newStandard
- Class\MCRFullTextSearchRefresh.run
- Class\PdsRebateAgreementValidate.validate
- Class\PurchRFQFormLetter.main
- Class\ReqTransPoMarkFirm.getPurchIdSingleThread
- Class\TAM VendRebateCorrectClaims.correctClaims
- Class\TAM VendRebateCorrectClaims.createClaimCorrection
- Class\TAM VendRebateCorrectClaims.rebateAmountPerUnit
- Class\WhsReleaseToWarehouseForm.buttonRelease_clicked
- Classes\LedgerAllocationRules.ValidateDimension
- Classes\ProjJournalCheckPost.checkFeeJournalDimensions
- Commission_Sales.run
- CreditcardPaymentCardTokenize.getFromDialog
- CustInPaymDialog.openDialog
- CustVendDisputeHelper.canDeleteDispute
- EcoResCategoryTreeDatasource.new
- EcoResProductRelationTable.validateWrite
- Form\EcoResProductCreate.updateCallers
- Form\InventOnhandReserve\DataSource\InventSum.reserveNow
- Form\PdsRebateAgreement\DataSource\PdsRebateAgreement.executeQuery
- Form\ProcCategoryHierarchyManagement\FormDesign\CategoryTreeGroup\CategoryTreeCtrl.selection
- Form\ReqSupplyDemandSchedule.updateDesign
- Form\SalesTable\DataSource\MCRSalesLineDropShipment\field\DropShipment.modified
- Form\SalesTable\DataSource\SalesTable.create
- FormletterService.removeProforma
- InventJournalTrans.validateWrite
- InventJournalTrans_Tag.validateWrite
- InventMovement.addLedgerPhysicalAmount
- InventMovement.canAutoReserveQuantity
- InventTransSerialNumberCreate.checkFormat
- InventUpd_Reservation.updateReserveLess
- LedgerJournalEngine.onSegmentChangedForPrimaryAccount
- ledgerJournalTransCusPaym.enableDisableMandate
- LedgerTransStatementDP.processOffsetAccountInStaging
- PdsBatchAttribReserveForm.checkReserveLine
- PriceDisc.findDiscAgreement
- priceDiscAdmCheckPost.postJournal
- PriceDiscHeading.updateDiscQty
- PriceDiscHeading.updateMultiLineDiscTmp
- PriceDiscPolicyFindOrCreate.run
- ProjInvoiceControl.projInvoiceControl
- ProjPostCostJournal.new
- PurchLineType.validateWrite
- ReqTransFormExplosion.tmpReqExplosionOnhandBuildServer
- ReqTransPoMarkFirm.createPurchTable
- ReqTransPoMarkFirm.updatePurchBuyerGroup
- RequisitionPurchaseOrderGeneration.createPurchaseOrder
- RetailBarCodeManagement.CreateBarCodeNoDim
- RetailTransactionServiceOrders.createCustomerOrder
- RetailTransactionTransformer.readTransactionSalesTrans
- SalesLine.createLine
- SalesLine.initFromPriceDisc
- SalesLine.insert
- SalesLine.update
- SalesLine.validateDelete
- SalesLine.writeRetailSalesLine
- SalesTable.updateMultiLineDisc
- SmaServiceFunctionLine_transfer.Run
- SmmOpportunityStatusUpdate.updateFromQuote
• Table\MCRCustpaymTable.salesTableByPassCreditLimit, displayOrderID, getCurrency, and mcrCustPaym\getCustomerPostingProfile
• Table\ReqPO.findAnySalesLineForReqPO
• TaxUncommitted.createTaxUncommitted, added local method createTaxUncommittedFromTmpTaxWorkTrans
• TmpTaxReport_IT.create
• TrvExpTrans.defaultTaxGroupFromWorker
• WHSBillOfLadingDP.insertWHSBillOfLadingTmp
• WHSControlItemId.populate
• WhsWarehouseRelease.creditLimitCheck
• WhsWorkCreate.addRangesToWorkTemplateQuery
• WHSWorkExecute.CreateTransferJournalLine
• WhsWorkTypePrintHandler.buildLabelAndConfirm

Other extensibility enhancements

• The PriceDiscHeading map was made extensible.
• **Retail channel:** Pre-triggers were added for Shipped, PackingSlip, and MarkAsPacked.
• **Retail channel:** The Cancellation charge dialog box can be overridden.
• **Retail channel:** Recall order default parameter value extension for the search order dialog box.
This topic lists the extensibility features that were implemented in Microsoft Dynamics 365 for Finance and Operations version 10.0.2. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Enumerations made extensible

The following enumerations have been made extensible in this update:

- MarkupModuleType
- MCRCustPaymType
- PaymSchededBy

SQL operations made extensible

The following SQL operations have been made extensible in this update:

- JmgPayAdjustment.payAdjustLoop
- ProjPosting.ExtensionHash.New field
- WmsArrivalOverviewGeneration.buildPurch
- WmsArrivalOverviewGeneration.buildTransferOrder

Metadata changes

The following metadata changes have been made in this update:

- CostSheetPercent.NoOfDecimalsIsExtensible
- WHSCycleCountingWarehouseWorkLineEntity.IsPublic

Refactored methods

The following methods have been refactored to support extensibility:

- /Forms/ProjJournalTable/datasource/ProjJournalTable.initValue
- /Forms/PurchReqTable.instantiatePurchReqTableForm
- /Forms/PurchReqTable/DataSource/PurchReqTable.init
- /Forms/SalesQuotationProjLinkWizard/Controls/ProjInvoiceId.lookup
- /Tables/SalesTable.lastQuotation
- AccPolicyProductReceipt.isAccountingRequiredForSourceDocLine
- AssetFixedAssetEntity.overrideDataSource
- AssetProposalDepreciation.run
- AssetTableMethod.init
- BankAccountReconcile.validate
- Class\BomCalcCost.calcCostModel
- Class\MCRLoadContinuityCustInfo.insertLineData
- Class\McrPriceHistoryUpdate.insertNewlyFoundReferences
- Class\McrPriceHistoryUpdate.update
- Class\McrPriceHistoryUpdate.updatePriceHistoryLineReferences
- Class\ProjCopyItemEstimates.copyToItemRequirment
- Class\PurchAutoCreate_RFQ.createPurchOrderRFQLineReference
- Class\ReqEventProcessDeleteUnusedKanban.deleteUnusedKanban
- Class\ReqEventProcessDeleteUnusedKanban.run
- Class\ReqTransUpdate.updateLogAddQty
- Class\SalesCancelOrder.run
- Class\SalesCreateOrderFromCustomer.main
- Class\TAMVendRebateCorrectClaims.rebateAlreadyGiven
- Class\TamVendRebateTableStatusType_Approved.runPayment
- Class\TamVendRebateTableStatusType_Calculated.inserted
- Class\TamVendRebateTableStatusType_Calculated.runPayment
- Classes\TaxWithhold.createAllTaxWithholdTrans
- Classes\TaxWithhold.isCalculateTaxWithholdingNeeded_TH
- Classes\TaxWithhold.postTaxWithhold
- Classes\TaxWithhold.totalInvoiceLineAmountSettled_TH
- CustDirectDebitMandate.setDefaultMandate
- CustDueReportDetailDP.class declaration
- CustDueReportDetailDP.insertCustDueReportDetailTmp
- CustQuotationConfirmJour.printJournal
- CustVendCreatePaymJournal.pack
- CustVendCreatePaymJournal.parmHasBatchBeenSplit
- CustVendEditTaxBranch_TH.init
- CustVendSumForPaym.run
- CustVendTransSettlement.post
- DimDerDistRuleSalesComplInvoice_BR.createDimAllocForProjRevenue
- EcoResProductCreateExtended.SetAllowEditField
- EcoResProductVariantEntity.findDataSource
- FBSpedFileCreator_Fiscal_BR.createRecordC195
- Form\ProdTableCreate.canContinueWithEmptyDim
- Form\PurchCreateFromSalesOrder\DataSource\SalesLine.included
- Form\PurchCreateFromSalesOrder\DataSource\SalesLine.specifyVendAccount
- Forms\TaxWithholdTable.init
- FreeTextInvoiceDP.setSysDocuBrandDetails
- InventItemOrderSetupMap.checkNotStopped
- InventNonConformanceTable.InventNonConformanceTable.Create
- InventUpd_Estimated.updateAutoDimBatchId
- InventUpdateReserveMore.InventUpdateReserveMore
- InventValueReportPopulateItem.updateReportLinePL
- JmgCalcApproveDateView.viewDate member
- JmgCalcApproveWeekView.viewDate member
- JmgPayAdjustment.payAdjustLoop
- LedgerJournalPeriodicCopy.journalTransCopy
- LedgerTransStatementDP.populateTempTableLedgerInStaging
- MCRCustpaym.validateWrite
- MCRFullTextSearch.buildSearchText
- MCRFullTextSearch.truncate
- MCRHoldCodeTrans.insert
- MCRHoldCodeTrans.setOrderStoppedFlag
- MCRHoldCodeTrans.unreserve
- McrPriceHistoryForm.calcPotential
- McrPriceHistoryForm.insertPotentialTradeAgreements
- PaymTerm.validateWrite
- PdsRebateAgreement.checkLineBreaks
- PdsRebateAgreement.groupChangeCheckValid
- PdsRebateAgreement.lineAmountHasGapOrOverlap
- PdsRebateAgreement.lineQuantityHasGapOrOverlap
- PdsRebateAgreementLine.selectRebateAgreementLineMax
- PriceDisc.mcrCalcPostageDisc
- PriceDiscLinePolicyRule.retrieveSystemPolicyFieldList
- ProdUpdCostEstimation.updateSubPurchLine
- ProjBudgetManager.createBudgetLineDetail
- ProjBudgetManager.getQuery
- ProjForecastCost.validateWrite
- ProjForecastEmpl.validateWrite
- ProjForecastRevenue.validateWrite
- ProjLedgerUpdate.insert
- ProjPlanVersionsManager.importHierarchy
- ProjPlanVersionsManager.importProjPlanVersionRecords
- ProjPost.PostCost
- ProjPost.PostCost
- ProjWorkBreakdownStructureHelper.addQuotationRelatedRecordsForTask
- ProjWorkBreakdownStructureHelper.Addtask
- ProjWorkBreakdownStructureHelper.Addtask
- ProjWorkBreakdownStructureHelper.Addtask
- ProjWorkBreakdownStructureV2FormHelper.IndentTaskV2
- ProjWorkBreakdownStructureV2FormHelper.MoveTasks
- PurchFormletterParmDataInvoice.createParmLinesAndTable
- PurchLine.delete
- PurchLine.distributionUpdateNeeded
- PurchLine.initFromPriceDisc
- PurchLine.insert
- PurchLine.update
- PurchLineType.statusChangeAllowed
- ReqEventProcessKanban.newStandard
- ReqTransNeutralTracker.trackReqTrans
- ReqTransPoMarkFirm.create
- Retail channel: CartWorkflowHelper.AllowAggregation
- RetailEcoResProductReleaseManager_Extension.setAndSaveRetailProductProperties
- RetailMassUpdateUploadDBManager.insertIntoProductProperty
RetailPeriodicDiscount.validatePriceGroup
RetailTransactionServiceCustomer.newCustomer
RetailTransactionTransformer.ReadDiscountLines
SalesInvoiceDP.SetSysDocuBrandDetails
SalesInvoiceJournalPost.run
SalesInvoiceJournalPostBase.run
SalesLine.CheckItemId
Table\InventTable.purchPriceAgreement
Tables\TaxWithholdTrans.copyTaxWithholdTrans, initFromTaxWithholdTable, insert, validateWrite, amountTotalWHT, existPeriod_TH
TAMVendRebatePaymentPost.main
TAMVendRebateTableProcess.runProcess
TrvPostExpenseHeader.postCustVendTransactions
TrvPostExpenseHeader.postCustVendTransactions
WhsControlLicensePlateId.process
WhsLicensePlateLabelBuild.insertSingleLabelMenuItem
WhsLicensePlateLabelBuild.insertSingleLabelPrintLine
WhsrfControlData.processLegacyControl
WhsWorkCreateProdPut.createReportFinishedParameters
WhsWorkCreateProdPut.insertProdParmForCoByProduct
WhsWorkCreateProdPut.insertProdParmForProdItem
WhsWorkCreateProdPut.setAcceptError
WHSWorkCreateReplenishment.checkExistingReplenWork
WHSWorkExecuteDisplay.buildPick
whsWorkExecuteDisplayInquiryLocation.buildLocationInquiry
WmsArrivalOverviewGeneration.buildInventTransId
WmsOrderTransType_OutputDontPostTransfer.decreaseQty
WmsOrderTransType_OutputDontPostTransfer.increaseQtyOverdelivery

Other extensibility enhancements

- The accessModifier of Classes\BankPositivePayExport.Class changed from private to protected.
- The InventItemOrderSetupMap map was made extensible.
- **Retail channel:** Custom columns in RetailTransactionView.
- **Retail channel:** The sign-in request can be overridden.
- **Retail channel:** Shipping view extension controller class.
- **Retail channel:** Support for the AppBar button in AddressAddEditView.
- **Retail channel:** Support for overriding the Bank deposit amount key in the dialog box.
This topic lists the extensibility features that were implemented in Microsoft Dynamics 365 for Finance and Operations version 10.0.1. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

### Enumerations made extensible

The following enumerations have been made extensible in this update:

- ACOCostStatus_BR
- ACOCostType_BR
- ACOJournalType_BR
- BankModuloCheck_NO
- InventTransferOrderType_BR
- ProdJourType
- ProjTransStatus
- RetailLabelTypeBase
- RetailLedgerBank
- RetailTenderFunction
- SalesPurchTrntype_BR
- SMAGetPriceFrom
- SMASubscriptionIndexChange

### SQL operations made extensible

The following SQL operations have been made extensible in this update:

- InventSumDelta.findInventSumDeltaInventSumFieldsAll.
- LedgerFiscalJournal was changed so that it uses QueryObject.
- TaxTransDP

### Metadata changes

The following metadata changes have been made in this update:

- Data Entities/WMSItemArrivalJournalLineEntity.IsPublic, PublicCollectionName, PublicEntityName.
- DataEntities/LedgerJournalNameEntity/Fields/voucherSeriesCode.Allow Edit, Allow Edit on Create.
- DataEntities/LedgerJournalNameEntity/Fields/VoucherSeriesCompanyId.AllowEdit.
- Enums\SalesStatus::Backorder, Delivered.Label.
- Extended Data Types/WeightBase.Scale.
- InventTransferOrders added a form control group in the grid.

### Refactored methods
The following methods have been refactored to support extensibility:

- AssetProposalDepreciation.run
- BankStatementValidate.validateDate
- Class\BankDocument\BankAccountTrans.loadSourceBuffer
- Class\BankReconMatchingRuleAutoProcessor.doProcessMatchRule
- Class\ProjJournalTransMapForm.initFromProjTable
- Class\RetailEodStatementPaymentJournal.createPaymentJournalLine
- Class\RetailKitAssemblyOrder.CreateOrUpdateBOMJournal
- Class\RetailTransactionServiceOrders.createOrUpdateRetailOrderLines
- Class\SalesInvoiceController.initReportName_IN
- Class\SalesInvoiceJournalPost.endUpdate
- Class\WrkCtrScheduler.loadRoute
- CreditCard.mcrInitFromCustPaymTable
- CreditcardProcess.mcrDoCapture
- CreditCardProcess.mcrDoRefund
- CreditCardProviderProcess.Submit
- CustCollectionLetterCreate.skipCustomer
- CustVendCheque.output
- CustVendSumForPaym.run
- EFDOCDanfe_BR.additionalInformationPageBreak
- EfDocDANFEDP_BR.additionalInformationBox
- ERDocuManagement.insertFile
- ERFileDestinationAttachment.saveFile
- ERFileDestinationBrowser.saveFile
- Form\BankReconciliationWorksheet.Init
- FreeTextInvoiceController.initReportName_IN
- HcmWorkerTransition.createHcmWorker
- InventCostClosingCancel_Init.createTasks
- InventDimCtrl_Frm_OnHand.initFromCaller
- InventMov_Jour_BOM.journalPostTrans
- InventProcessGuideDisplayLicensePlateDetailsPageBuilder.generateItemInfoForLicensePlate
- InventProcessGuideDisplayLocationDetailsPageBuilder.generateItemInfoForLocation
- InventStockCardDP.createInventStockCardTmpLineDetail
- InventTable.checkProjCategoryId
- InventUpd_WHSReservation.updateReserveMore
- JmgCalcApproveWeekView.initializeData
- LedgerConsolidate.Run and getSelectedDimensionAttributes
- LedgerFiscalJournalDP_IT.addStarsToTmpTable
- LedgerFiscalJournalDP_IT.insertLedgerFiscalJournalTmp_IT
- LedgerFiscalJournalDP_IT.processReport
- LedgerJournalTrans.checkAllowEditWhenCheckPrinted
- LedgerJournalTransUpdateVend.pdateNow
- LedgerVoucherTransList.First
- LedgerVoucherTransList.next
- MCRFullTextIndexField.tableIdFromEnum
- MCRFullTextIndexField.viewFromTable
- MCRIInventSearch.searchProduct
- McrPriceHistoryLine_Purch.initAndInsertRebate
- PartyProvider.operatingUnitTypeToName
- PdsRebateAgreementValidate.construct
- POS_IssueLoyaltyCardView.NA
- PriceDiscAdmCheckPostPriceDiscTableUpdater.formattedQueryValue
- PriceDiscAdmTrans.checkItemRelation
- ProjBudgetManager.deleteProjBudgetLinesWhenZeroAmount
- ProjBudgetManager.updateProjBudgetLinesWithAmt
- ProjHourCostPrice.psaFindCostPrice
- ProjInvoiceProposalListPageInteraction.initializeQuery
- ProjPostEmplProposalSale.new
- ProjPostRevenueProposalSale.new
- ProjTable.initProjectFromCustomerAndInvoice
- ProjTransferPrice.findByContractResourceCategory, findTransferPrice, find
- ProjValElementServer.addProjToResource
- ProjValElementServer.deleteProjFromResource
- PurchPackingSlipJournalPost.postMarkupOnTrans
- PurchReqLine.setProjSalesPrice
- PurchRFQSendJournalCreate.createOrUpdateRFQLine
- ReqCalc.covCalcDim
- ReqCalc.covCalcDim
- ReqCalc.covCalcDim
- RequisitionPurchaseOrderGeneration.create
- RequisitionPurchaseOrderGeneration.create
- Retail extension point in the Commerce runtime (CRT) to override the ValidateCartLineQuantityAndPriceSymbol method
- RetailCatalogProductAttributeFormHelper.addProductAttributeControls
- RetailCreateSpecificLabel.makeLabel
- RetailEodStatementCustomerOrderInvoiceController.run
- RetailEodStatementPaymentJournal.ledgerBank2LedgerJournalACType
- RetailEodStatementPaymentJournal.postPaymentJournalForOthers, postPaymentJournalForSales, createTenderedPaymentLines, createPaymentJournalLine
- RetailEodTransactionTransformer.ReadTransactionHeader
- RetailEodTransactionTransformer.setExtensionProperty
- RetailEventNotificationAction.packingSlipCompletion
- RetailMediaAssociationHelper.associateProduct
- RetailProductPropertyManager.validateWriteOnInventModelGroupItem
- RetailSMBSeedGenerator.AccountReceivable
- RetailStatementPost.createPaymentLedgerTrans
- RetailTransactionSalesTransMark.MarkTransactions
- RetailTransactionServiceOrders.settleCustomerOrder
- RetailTransactionServiceOrders.cancelCustomerOrder
- RetailTransactionServiceOrders.createCustomerOrder
- RetailTransactionServiceOrders.createLedgerJournalForStore
- RetailTransactionServiceOrders.createOrUpdateRetailOrderHeader
Other extensibility enhancements

- **Retail channel**: Allow for extensions to support Select all and Clear all in OrderFulfillmentView.
- **Retail channel**: Expose Add return line to cart from the transaction application programming interface (API) by line ID.
- **Retail channel**: Line item locations can be viewed in OrderFulfillmentView.
- **Retail channel**: OrderFulfillmentView adds ICustomListColumn to allow for more information.
- Retail statement posting method adds another aggregation view by using the new RetailTransactionAggregationFieldList table that adds additional fields.
This is a list of extensibility features that were implemented in Dynamics 365 for Finance and Operations version 10.0. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

### Enumerations made extensible

These enumerations have been made extensible in this update.

<table>
<thead>
<tr>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssetAccrualCalendar</td>
</tr>
<tr>
<td>AssetYear</td>
</tr>
<tr>
<td>BankReconciliationReportType</td>
</tr>
<tr>
<td>BudgetPlanColumnPeriodLength</td>
</tr>
<tr>
<td>BudgetPlanHCMReportGroupOption</td>
</tr>
<tr>
<td>CurrencyTypeBrief_RU</td>
</tr>
<tr>
<td>EInvoiceStatus_IT</td>
</tr>
<tr>
<td>EInvoiceStatus_IT</td>
</tr>
<tr>
<td>HRPAuthorityBasis</td>
</tr>
<tr>
<td>HuExchOutflowType</td>
</tr>
<tr>
<td>InventJournalTagStatus</td>
</tr>
<tr>
<td>InvoiceAssociationType</td>
</tr>
<tr>
<td>MCRClaimType</td>
</tr>
<tr>
<td>MCRMerchandisingEventCategory</td>
</tr>
<tr>
<td>PaymAttribute</td>
</tr>
<tr>
<td>PaymProposalReportedBy</td>
</tr>
<tr>
<td>PayrollCategory</td>
</tr>
</tbody>
</table>
SQL operations made extensible

These SQL operations have been made extensible in this update.

**OPERATION**

- JmgStampJournalTable.makeLines
- MCRDropShipStatusUpdate_PurchLine.updatePurchDropShipStatusOnRecord
- MCRDropShipStatusUpdate_PurchTable.updatePurchDropShipStatusOnRecord
- SalesInvoiceJournalCreate.checkDocumentData_PL
**Metadata changes**

These metadata changes have been made in this update.

<table>
<thead>
<tr>
<th>Metadata changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>/Data Model/Data Entities/BOMBillOfMaterialsVersionV2Entity.IsPublic</td>
</tr>
<tr>
<td>/Data Model/Data Entities/InventItemBatchEntity.IsPublic</td>
</tr>
<tr>
<td>/Data Model/Data Entities/InventProductSpecificOrderSettingsV2Entity.IsPublic</td>
</tr>
<tr>
<td>/Data Model/Data Entities/InventQualityGroupItemAssignmentEntity.IsPublic</td>
</tr>
<tr>
<td>/Data Model/Data Entities/InventQualityTestGroupEntity.IsPublic</td>
</tr>
<tr>
<td>/Data Model/Data Entities/ProductionPoolEntity.IsPublic</td>
</tr>
<tr>
<td>/Data Model/Data Entities/WMSItemArrivalJournalHeaderEntity.IsPublic, PublicCollectionName, PublicEntityName</td>
</tr>
<tr>
<td>/DataModel/Tables/WMSStorageLoadUnitReqTrans.WMSStorageLoadUnitReqTran</td>
</tr>
<tr>
<td>DimensionHierarchyType/EnumValue/RDeferrals</td>
</tr>
<tr>
<td>AOT/Data Model/Tables/CategoryTable.Create RecId Index</td>
</tr>
<tr>
<td>EcoResProductSpecificUnitOfMeasureConversionEntity.Property.IsPublic</td>
</tr>
<tr>
<td>InventProductSpecificOrderSettingsV2Entity.Property.IsPublic</td>
</tr>
<tr>
<td>&quot;No of Decimals is Extensible&quot; property on several EDTs</td>
</tr>
<tr>
<td>RetailLoyaltyRewardPoint.Replacement Key</td>
</tr>
<tr>
<td>Tables/CustTrans/Relations/ThirdPartyBankAccountId.Validate</td>
</tr>
<tr>
<td>Tables/EInvoicePropertyTable/Relations/EInvoicePropertyTypeTable.RelationshipType</td>
</tr>
<tr>
<td>Tables/ResourceSetup.FormRef</td>
</tr>
<tr>
<td>Tables/WHSTmpWorkExecuteListBoxItems/Fields/Elements.EDT</td>
</tr>
</tbody>
</table>

**Refactored methods**
These methods have been refactored to support extensibility.

**REFACTORED METHODS**

- AdvancedLedgerEntryLine.setProjInvoiceLineLedgerDimension
- AgreementConfirmationDP.getSalesAgreementHeader
- AgreementConfirmationDP.getSalesAgreementHeaderHistory
- PurchAutoCreate_Sales.createPurchLine
- PurchCreateFromSalesOrder.run
- InventItemPrice.insert
- InventJournalTrans.setCostPrice
- PurchLine.initFromReqPO
- AssetBook.initDepreciationProfile
- AssetDepreciationProfile.validateStraightLine
- AssetProposalDepreciation.run
- BankPaymAdvicePrint.BankPaymAdvicePrint (variable)
- BankReconciliationMatchingMatchProcessor.constructMatch
- BankReconMatchingMatchStmtReversalDoc.Multiple
- BankStatementDocumentEntity.postGetStagingData
- BankVoucher.post
- BomCalcItemLine.mustExplodePrice
- BOMCopyToProd.delete
- BOMCreateDialog.promptCreateBOMDialog
- BomRouteCopyJob.initFromItemId
- BudgetPlanningConfiguration.displayYearOffset
- BudgetPlanningConfiguration.updateColumnPeriodLengthValueLabel
- CatVendorCatalogProductApproval.getApprovedProductForRetail
- LedgerJournalTransType.validateAccountType
- LedgerTransferOpening.processQuery
### REFACTORED METHODS

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BankPositivePayExport.generatePositivePayFile</td>
<td></td>
</tr>
<tr>
<td>BankPositivePayExport.updateBankPositivePay</td>
<td></td>
</tr>
<tr>
<td>CaseSendEmail.getEmailMessage</td>
<td></td>
</tr>
<tr>
<td>ContactPerson.insert</td>
<td></td>
</tr>
<tr>
<td>CostSheetModeStrategyStaging.createCostSheetNodes</td>
<td></td>
</tr>
<tr>
<td>CreditCard.recordAuthorization</td>
<td></td>
</tr>
<tr>
<td>CreditCard.recordCapture</td>
<td></td>
</tr>
<tr>
<td>CreditCardPaymentJournal.createJournal</td>
<td></td>
</tr>
<tr>
<td>CreditCardPaymentJournal.Init</td>
<td></td>
</tr>
<tr>
<td>CreditCardPaymentJournal.run</td>
<td></td>
</tr>
<tr>
<td>CustAgingReportContract.Validate</td>
<td></td>
</tr>
<tr>
<td>CustAgingReportDP.CustAgingReportTmp</td>
<td></td>
</tr>
<tr>
<td>CustAgingReportDPClass.insertCustAgingReportTmp</td>
<td></td>
</tr>
<tr>
<td>CustAgingReportDPClass.setCustAgingReportTmpInReverse</td>
<td></td>
</tr>
<tr>
<td>CustBalanceList.insertIntoTmpAccountSumV2</td>
<td></td>
</tr>
<tr>
<td>CustBillOfExchangePostRemit.postSettlingStep</td>
<td></td>
</tr>
<tr>
<td>CustCollectionsSetTransactionStatusHelper.createActions</td>
<td></td>
</tr>
<tr>
<td>CustCustomerBaseEntity/CustCustomerEntity/CustCustomerV2Entity/CustCustomerV3Entity.processChangesForApproval</td>
<td></td>
</tr>
<tr>
<td>CustCustomerDetailEntity/CustCustomerDetailV2Entity.processChangesForApproval</td>
<td></td>
</tr>
<tr>
<td>CustInvoiceJour.setInvoiceAddress</td>
<td></td>
</tr>
<tr>
<td>CustInvoiceLine.getCustBillingCodeLedgerAccount</td>
<td></td>
</tr>
<tr>
<td>CustInvoiceLine.setProjInvoiceLineLedgerDimension</td>
<td></td>
</tr>
<tr>
<td>CustInvoiceLine.setProjInvoiceLineLedgerDimensionBase</td>
<td></td>
</tr>
<tr>
<td>CustInvoiceLine.shouldDefaultLedgerDimensionFromProject</td>
<td></td>
</tr>
<tr>
<td>REFACTORED METHODS</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>CustOutPaymRecord_Cheque.checkValues</td>
<td></td>
</tr>
<tr>
<td>CustPostInvoiceJob.custPostInvoiceUpdate</td>
<td></td>
</tr>
<tr>
<td>CustVendAgingCalculation.process</td>
<td></td>
</tr>
<tr>
<td>CustVendChequeSlipTextCalculator.getChequeDocLength</td>
<td></td>
</tr>
<tr>
<td>CustVendChequeSlipTextCalculator.getMinimumSlipLines</td>
<td></td>
</tr>
<tr>
<td>CustVendChequeSlipTextCalculator.fillSlipText</td>
<td></td>
</tr>
<tr>
<td>CustVendChequeSlipTextCalculator.Property</td>
<td></td>
</tr>
<tr>
<td>CustVendEditTaxBranch_TH.init</td>
<td></td>
</tr>
<tr>
<td>CustVendOutPaym.getSumByCurrency</td>
<td></td>
</tr>
<tr>
<td>CustVendPaymInvoiceWithJournal.createJournal</td>
<td></td>
</tr>
<tr>
<td>CustVendPaymInvoiceWithJournal.createPayment</td>
<td></td>
</tr>
<tr>
<td>CustVendPaymProposal.resolvePaymAccountAndType</td>
<td></td>
</tr>
<tr>
<td>CustVendPaymProposalLine.paymTransactionAmountMST</td>
<td></td>
</tr>
<tr>
<td>CustVendPaymProposalTransferToJournal.getVoucherNum</td>
<td></td>
</tr>
<tr>
<td>CustVendReversePosting.restoreCustVendTransOpen</td>
<td></td>
</tr>
<tr>
<td>CustVendSettle.postDueToAndFromCreateTrans</td>
<td></td>
</tr>
<tr>
<td>CustVendSettle.postExchRateLedgerTrans</td>
<td></td>
</tr>
<tr>
<td>CustVendSettle.settleNow</td>
<td></td>
</tr>
<tr>
<td>CustVendSettle.updateCustTaxInvoice_TH</td>
<td></td>
</tr>
<tr>
<td>CustVendSumUpJournal.createTrans</td>
<td></td>
</tr>
<tr>
<td>CustVendSumUpJournal.createVoucher</td>
<td></td>
</tr>
<tr>
<td>CustVendTransreorg.end</td>
<td></td>
</tr>
<tr>
<td>CustVoucher.updateProjTransPosting</td>
<td></td>
</tr>
<tr>
<td>DimDerDistRuleProjectRevenueExt.processRegularTransactions</td>
<td></td>
</tr>
<tr>
<td>DimDerDistRuleProjectRevenueExt.processIntercompanyTransCustInvoice</td>
<td></td>
</tr>
<tr>
<td>REFACTORED METHODS</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>DimDerDistRuleProjectRevenueExt.processIntercompanyTransExpense</td>
<td></td>
</tr>
<tr>
<td>DimDerDistRuleProjectRevenueExt.processIntercompanyTransTimesheet</td>
<td></td>
</tr>
<tr>
<td>DimDerJourRuleProjectTimesheetsExt.getDefaultDimensionAllocation</td>
<td></td>
</tr>
<tr>
<td>EcoResEnumerationAttributeTypeValue.createAttributeValuesFromEnum</td>
<td></td>
</tr>
<tr>
<td>EcoResProductReleaseForm.addProductsToRelease</td>
<td></td>
</tr>
<tr>
<td>EInvoice_IT.newCustInvoice</td>
<td></td>
</tr>
<tr>
<td>EInvoice_IT.newProjInvoice</td>
<td></td>
</tr>
<tr>
<td>EUSalesListReportingEngine.Construct</td>
<td></td>
</tr>
<tr>
<td>FiscalDocument_BR.lastIssueDateForSeries</td>
<td></td>
</tr>
<tr>
<td>FormletterJournalPost.docuRefCopyByRecId</td>
<td></td>
</tr>
<tr>
<td>ForecastSales.Update</td>
<td></td>
</tr>
<tr>
<td>HcmActionState.lookupReferenceActionTypeSetup</td>
<td></td>
</tr>
<tr>
<td>HcmWorkerinit</td>
<td></td>
</tr>
<tr>
<td>HcmWorker.updateEmploymentControls</td>
<td></td>
</tr>
<tr>
<td>HcmWorkerActionHireCompletion.getHrmApplication</td>
<td></td>
</tr>
<tr>
<td>HcmWorkerTransition.createHcmEmployment</td>
<td></td>
</tr>
<tr>
<td>HRCCompGridView.initCompRecord</td>
<td></td>
</tr>
<tr>
<td>HRMCompFixedEmpl.enforcePayRateTolerance</td>
<td></td>
</tr>
<tr>
<td>HRPDefaultSigningLimitRule.insertFormDataSourceJobDetail</td>
<td></td>
</tr>
<tr>
<td>HRPDefaultSigningLimitRule.populateDetailGrid</td>
<td></td>
</tr>
<tr>
<td>HRPDefaultSigningLimitRule.SaveValidation</td>
<td></td>
</tr>
<tr>
<td>HRPDefaultSigningLimitRule.insertOrUpdateFormDataSource</td>
<td></td>
</tr>
<tr>
<td>HRPDefaultSigningLimitRuleCompensation.getSelectedCompensation</td>
<td></td>
</tr>
<tr>
<td>HRPDefaultSigningLimitRuleCompensation.getAvailableCompensation</td>
<td></td>
</tr>
<tr>
<td>HRPDefaultSigningLimitRuleCompensation.selectRecords</td>
<td></td>
</tr>
</tbody>
</table>
REFACTORED METHODS

- HRPDefaultSigningLimitRuleCompensation.unselectRecords
- HrpWorkerLimit.getActiveDefaultSLRule,
- HrpWorkerLimit.getDefaultSigningLimits
- HrpWorkerLimit.getWorkerSigningLimit
- HrpWorkerLimit.getSigningLimitsIfRequestNotRequired
- InterCompanyTransferInventDim.Entire class
- InterCompanyTransferInventDim.transfer
- InventBatch.update
- InventCountCreate_Base.createInventJournalTrans
- InventInventoryDimensionEntityFieldsMapping.resolveInventDim
- InventMov_Jour_BOM.journalCheckTrans
- InventMov_Jour_Loss_Project.checkAccountOperations
- InventMov_Journal.journalSetItemId
- InventMov_Statement.pdsCWRemainPhysical
- InventMovement.performFinancialLedgerUpdate
- InventProcessGuideAdjustInController.initialStepName
- InventQualityManagementBlock.run
- InventQualityManagementCreateHandler.purchFormLetterBeforeHelper
- InventQualityOrderTableValidator.checkQty
- InventSum.retrieveMatchingInventSumDeltaForTTSid()
- InventTrackingRegisterTransForm.construct
- InventTransAdjust.updateNow
- InventTransferUpdReceive.updateInventTransferLine
- InventTransWms_Register.updateInventFromMovementServer
- InventUpd_ChildReference updateLess* methods
<table>
<thead>
<tr>
<th>REFACTORED METHODS</th>
</tr>
</thead>
<tbody>
<tr>
<td>InventUpd_ChildReference.updateMoreIssue</td>
</tr>
<tr>
<td>InventUpd_Estimated.updateAutoDimMovement</td>
</tr>
<tr>
<td>InventUpd_Physical.UpdatePhysicalReturnedIssue</td>
</tr>
<tr>
<td>InventUpd_Physical.updatePhysicalReturnedReceipt</td>
</tr>
<tr>
<td>InventUpd_WHReservation.continueInventTransUpdateReserveMoveLoop</td>
</tr>
<tr>
<td>InventUpdateOnhand.checkOnhand()</td>
</tr>
<tr>
<td>InventUpdateReserveMore.buildQueries</td>
</tr>
<tr>
<td>JmgMESDocuHandling.openFile</td>
</tr>
<tr>
<td>JmgProfiles.insertTimeGapsPlannedAbs</td>
</tr>
<tr>
<td>JmgStampJournalCalculate.run</td>
</tr>
<tr>
<td>JmgStampJournalTransfer.cancelExecute</td>
</tr>
<tr>
<td>JmgStampJournalTransfer.cancelExecute</td>
</tr>
<tr>
<td>LeanCost_Init.execute</td>
</tr>
<tr>
<td>LedgerAllocationController.allocateAmounts</td>
</tr>
<tr>
<td>LedgerAllocationProcessRequest.createVoucherDestinations</td>
</tr>
<tr>
<td>LedgerJournalCheckPost.replaceTmpVoucher</td>
</tr>
<tr>
<td>LedgerJournalDeleteTransaction.deleteLedgerJournalTransRelated</td>
</tr>
<tr>
<td>LedgerJournalEngine.currencyModified</td>
</tr>
<tr>
<td>LedgerJournalPeriodicCopy.journalVoucherCopy</td>
</tr>
<tr>
<td>LedgerJournalTrans.initForCurrency</td>
</tr>
<tr>
<td>LedgerJournalTrans.validateWrite_Server</td>
</tr>
<tr>
<td>LedgerTransModule.insertTransactionList</td>
</tr>
<tr>
<td>LedgerTrialBalanceContract.DataMemberAttribute</td>
</tr>
<tr>
<td>LedgerVoucherObject.allocateTransaction</td>
</tr>
<tr>
<td>LedgerAllocationController.allocateRecursive</td>
</tr>
<tr>
<td>REFACTORED METHODS</td>
</tr>
<tr>
<td>---------------------</td>
</tr>
</tbody>
</table>
| MCRCheckHoldWB
clicked               |
| MCROrderEventSetup.
find                        |
| MCRSalesOrderRecap.
Control:SubmitButton.
clicked                 |
| MCRSalesQuickQuote.
Modified                   |
| MCRImpPickingWorkbench
Trans.initFromSession.
Criteria                |
| MultilineString.POSDe
veloperSupport             |
| OriginalDocuments.
insertDocument                |
| PaymSchedCalc_Amount.
createTransaction            |
| PdsRebateFindAndCreate.
findPdsRebateAgreement.
AndCreateClaim                  |
| PdsRebateFindAndCreate.
findPdsRebateAgreement.
AndCreateClaim()                    |
| PdsRebatePaymentPost.
insertRebateEntry.
ForGrouping                     |
| PmfFormCtrl_BOM_BOMVersion.
movedFormulaSize            |
| PriceDiscAdmCheckPost.
postJournal                    |
| ProdJournalCheckPostProd.
postTransLedger                 |
| ProdJournalTransBOM.
inventBatchId.
validate                        |
| ProdMultiReportFinished.
insert                          |
| ProdUPDCostEstimation.
CreateProdBOM                   |
| ProdUpdReportFinished.
updateBOMConsumption          |
| ProdUpdStartUp.
createJournals                |
| ProjBegBalJournalTrans.
_CostSales.
validateField, validateWrite |
| ProjBudgetManager.
deleteBudgetLinesBefore.
ImportForRevs                   |
| ProjBudgetManager.
getQuery                      |
| ProjBudgetRevisionManager.
createBudgetLines              |
| ProjBudgetTransactionManager.
isOverrunAllowed            |
| ProjectCommitmentFacade.
updateProjectCommitmentsMap  |
REFACTORED METHODS

ProjectMainAccDimensionListProvider.populateMainAccountDimensionList

ProjForecastBudgetCopy.do_Cost

ProjForecastBudgetCopy.do_empl

ProjForecastBudgetCopy.do_onAcc

ProjForecastBudgetCopy.do_sales

ProjGroupChange.checkPostedTrxAccounts

ProjIntercompanyCustomerInvoiceCreator.createInvoiceLine

ProjInvoiceJournalPost.postCustVend

ProjInvoiceJournalPost.validateNoTax

ProjInvoiceProposalInsertLines.run

ProjJournalTrans.validateWrite

ProjPlanVersionCopyHierarchy.addProjPlanVersionFields

ProjPlanVersionCopyHierarchy.insertProjPlanVersionRecords

ProjPlanVersionCopyHierarchy.ProjPlanVersionCopyHierarchy

ProjPlanVersionsManager.importProjPlanVersionRecords

ProjPost.PostNeverLedger

ProjPost.PostTurnover

ProjPosting.updateDatasourceRanges

ProjTable.validateWrite

ProjTable.validateWriteServer

ProjTask.addTask


ProjWBSDataEntityHelper.postInsertOperation

PurchAutoCreate_ReleaseFromAgreement.createLines

PurchInvoiceJournalPost.calcLastPurchPrice
<table>
<thead>
<tr>
<th>REFACCTORED METHODS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PurchPackingSlipJournalPost.updateSourceLineBeforePosting</td>
</tr>
<tr>
<td>PurchReqLine.defaultBuyingLegalEntity</td>
</tr>
<tr>
<td>PurchRFQCaseAutoCreate_PurchReq.calcRFQHeaderValues</td>
</tr>
<tr>
<td>PurchTable/InventDim/InventBatchId.modified</td>
</tr>
<tr>
<td>ReqTransPoMarkFirm.executeAction</td>
</tr>
<tr>
<td>ReqTransPOMarkFirm.CreateProdBOM</td>
</tr>
<tr>
<td>ReqTransPoMarkFirm.purchTablePostProcessing</td>
</tr>
<tr>
<td>ReqTransPoMarkFirm.setDeliveryDateAndPriceDisc</td>
</tr>
<tr>
<td>ReqTransPoMarkFirm.setGroupingIndicators</td>
</tr>
<tr>
<td>RequisitionPurchaseOrderGeneration.Create</td>
</tr>
<tr>
<td>RequisitionPurchaseOrderGeneration.createPurch</td>
</tr>
<tr>
<td>RequisitionPurchaseOrderGeneration.getVendors</td>
</tr>
<tr>
<td>ResReserveCapacity.getCapacityPercentage</td>
</tr>
<tr>
<td>RetailCreateLinesFromProductsToAdd.loadDiscountLines</td>
</tr>
<tr>
<td>RetailMassUpdateValidator.validateWriteOnInventModelGroupItem</td>
</tr>
<tr>
<td>RetailMediaAssociationHelper.populateMediaAssociationTable</td>
</tr>
<tr>
<td>RetailOENInfo.parseEmailTemplate</td>
</tr>
<tr>
<td>RetailPrintLabels.loadFromArgs</td>
</tr>
<tr>
<td>RetailPrintLabels.loadLines</td>
</tr>
<tr>
<td>RetailTransactionServiceOrders.createOrUpdateRetailOrderHeader</td>
</tr>
<tr>
<td>RetailTransactionServiceOrders.createOrUpdateRetailOrderLines</td>
</tr>
<tr>
<td>SalesConfirmJournalPost.createReportData</td>
</tr>
<tr>
<td>SalesFormLetter_Invoice.checkInvoicePrices</td>
</tr>
<tr>
<td>SalesInvoiceDP.setPackingSlipDetails</td>
</tr>
<tr>
<td>SalesInvoiceJournalPostBase.updateInventory</td>
</tr>
<tr>
<td>REFACTORED METHODS</td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td>SalesInvoiceJournalPostBase.updateInventoryFinancialForSalesInvoiceLine</td>
</tr>
<tr>
<td>SalesLine.CheckItemid</td>
</tr>
<tr>
<td>SalesLine.getInventQtyFromCWUnit</td>
</tr>
<tr>
<td>SalesLine.setInventDeliverNow</td>
</tr>
<tr>
<td>SalesLineType.validateWrite</td>
</tr>
<tr>
<td>SalesQuotationDP.createTaxLines</td>
</tr>
<tr>
<td>SalesQuotationDP.itemid</td>
</tr>
<tr>
<td>SalesQuotationLine.PriceDate</td>
</tr>
<tr>
<td>SalesQuotationTable.active</td>
</tr>
<tr>
<td>SalesTable-DataSource_mcrSalesTable-DataField_Sourceld.modified</td>
</tr>
<tr>
<td>ShipOrderForm.POS.ChangeOriginOnShipOrders</td>
</tr>
<tr>
<td>SmabomDesignerCtrl.listInsertHistory</td>
</tr>
<tr>
<td>SmabomDesignerCtrl.treeSubstituteBOMonNode, treeDeleteNode, treeDeleteChildrenCollect</td>
</tr>
<tr>
<td>SMAServiceObjectrelation.jumpRefBOMTable</td>
</tr>
<tr>
<td>SubledgerJournalizerProjectExtension.createProjectActualCostDetail</td>
</tr>
<tr>
<td>SubledgerJournalizerProjectExtension.createProjectActualSalesDetail</td>
</tr>
<tr>
<td>SubledgerJournalTransferCommand.insertGeneralJournalAccountEntryRelated</td>
</tr>
<tr>
<td>SubledgerJournalTransferCommand.insertGeneralJournalAccountEntryRelatedDetail</td>
</tr>
<tr>
<td>SubledgerJournalTransferCommand.insertGeneralJournalAccountEntryRelatedDetail</td>
</tr>
<tr>
<td>SubledgerJournalTransferCommand.insertGeneralJournalAccountEntryRelatedSummarized</td>
</tr>
<tr>
<td>SubledgerJournalTransferCommand.insertGeneralJournalAccountEntryRelatedSummarized</td>
</tr>
<tr>
<td>SubledgerJournalTransferCommand.insertGeneralJournalEntryRelated</td>
</tr>
<tr>
<td>SuppItem.calcSuppItem</td>
</tr>
<tr>
<td>TaxProformaSpec.parmTaxSpec</td>
</tr>
<tr>
<td>TaxWithhold.postTaxWithhold</td>
</tr>
<tr>
<td>REFACTORED METHODS</td>
</tr>
<tr>
<td>--------------------</td>
</tr>
<tr>
<td>TaxWithholdSlipDP_TH.createTaxWithholdSlipTmp</td>
</tr>
<tr>
<td>TaxWithholdSlipDP_TH.createTaxWithholdSlipTmp</td>
</tr>
<tr>
<td>TmsProcessXML_Base.readRateShipment</td>
</tr>
<tr>
<td>TMSRouteHelper.getShipDates</td>
</tr>
<tr>
<td>TradeLineNumberManager.checkLineNumber</td>
</tr>
<tr>
<td>TrvCreditCardReminder.mail</td>
</tr>
<tr>
<td>TrvCreditCardReminder.runQT</td>
</tr>
<tr>
<td>TrvExpenditureParticipantProvider.resolveFromDimensions</td>
</tr>
<tr>
<td>TrvExpenditureParticipantProvider.resolve</td>
</tr>
<tr>
<td>TrvExpenditureParticipantProvider.resolveProjectAuthorities</td>
</tr>
<tr>
<td>TrvExpenses.openSplitDetailsForm</td>
</tr>
<tr>
<td>TrvExpTable.validateSubmit</td>
</tr>
<tr>
<td>VendAgingReportController.getReportName</td>
</tr>
<tr>
<td>VendBalanceList.insertIntoTmpAccountSum</td>
</tr>
<tr>
<td>VendInvoiceInfoListPage.postInvoice</td>
</tr>
<tr>
<td>VendOutPaymRecord_Cheque.checkValues</td>
</tr>
<tr>
<td>VendVendorEntity/VendVendorV2Entity.processChangesForApproval</td>
</tr>
<tr>
<td>VestingID.Table: HRMCompVarAward</td>
</tr>
<tr>
<td>WhsContainerization.packTmpWorkLine</td>
</tr>
<tr>
<td>WhsControlBatchId.process</td>
</tr>
<tr>
<td>WHSPostPackingSlip.canShipConfirm</td>
</tr>
<tr>
<td>WHSPostPackingSlip.shipConfirmLoad</td>
</tr>
<tr>
<td>WHSProcessGuideStartChangeWarehouseStep.doExecute</td>
</tr>
<tr>
<td>WhsrfControlData.batchExistInLocation</td>
</tr>
<tr>
<td>WhsShipConfirm.tmsMultiLoadShipConfirm</td>
</tr>
<tr>
<td>REFACTORED METHODS</td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td>WHSSplitWork.handleOriginalWorkLine</td>
</tr>
<tr>
<td>WHSSplitWork.handleRemainingPickTrans</td>
</tr>
<tr>
<td>WHSSplitWork.processRemainingTransaction</td>
</tr>
<tr>
<td>WHSSplitWork.updateClosedPickTrans</td>
</tr>
<tr>
<td>WhsUnShip.cleanUpTOInventTransDims</td>
</tr>
<tr>
<td>WhsWarehouseRelease.createShipmentsForTransferOrders</td>
</tr>
<tr>
<td>WHSWorkCreate.createWorkInventTrans</td>
</tr>
<tr>
<td>WHSWorkCreate.createWorkTable</td>
</tr>
<tr>
<td>WhsWorkCreateProdPut.createReportFinished</td>
</tr>
<tr>
<td>WhsWorkCreateReceiving.createBatch</td>
</tr>
<tr>
<td>WHSWorkExecute.putAwayToLocation()</td>
</tr>
<tr>
<td>WHSWorkExecuteDisplay.buildInventoryStatus</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplay.getNextFormState</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplay.processTrackingDimDetails</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplay.processVendorBatchDetails</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplay.processWorkLine</td>
</tr>
<tr>
<td>WHSWorkExecuteDisplay.setBatchDetails</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplayLoadItemReceiving.buildPOReceiving</td>
</tr>
<tr>
<td>WHSWorkExecuteDisplayLPReceiving.generateItemInfoForReceiving</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplayPOLineReceiving.buildPOReceiving</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplayPOLineReceiving.buildPOReceiving</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplayReportAsFinished.displayForm</td>
</tr>
<tr>
<td>WHSWorkTable.satisfyDemandWorkLine</td>
</tr>
<tr>
<td>WmsArrivalCreateJournal.createWMSJournalTransFromArrivalDetails</td>
</tr>
<tr>
<td>WmsJournalCheckPostReception.returnOrderUpdate</td>
</tr>
<tr>
<td>REFACTORED METHODS</td>
</tr>
<tr>
<td>---------------------------------------------------------</td>
</tr>
<tr>
<td>WmsOrderCreate.updateCreatewmsOrder</td>
</tr>
<tr>
<td>WorkflowHierarchyProviderHelperEventHandler.addDataSourceFieldsDelegate</td>
</tr>
<tr>
<td>WorkflowHierarchyProviderHelperEventHandler.loadLimits</td>
</tr>
<tr>
<td>WrkCtrScheduler.Prod.saveOperation</td>
</tr>
</tbody>
</table>

## Other changes

The following additional changes have been made for extensibility.

- Convert queries where InventSumFields is used to SysDa.
This is a list of extensibility features that were implemented in Dynamics 365 for Finance and Operations version 8.1.3. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Enumerations made extensible

These enumerations have been made extensible in this update.

<table>
<thead>
<tr>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttributeDataType</td>
</tr>
<tr>
<td>BankDocumentBookType</td>
</tr>
<tr>
<td>BudgetPlanHCMReportGroupOption</td>
</tr>
<tr>
<td>CommitmentType</td>
</tr>
<tr>
<td>CommitmentType</td>
</tr>
<tr>
<td>CustVendNegInstStatus</td>
</tr>
<tr>
<td>CzAdvanceInvoiceStatus</td>
</tr>
<tr>
<td>DateTransactionDuedate</td>
</tr>
<tr>
<td>LedgerAllocationMethod</td>
</tr>
<tr>
<td>LedgerCovDocumentType</td>
</tr>
<tr>
<td>MCRFraudType</td>
</tr>
<tr>
<td>PercentHours</td>
</tr>
<tr>
<td>PerDayWeekMthQtYr</td>
</tr>
<tr>
<td>PrepaymentHandlingLayout_W</td>
</tr>
<tr>
<td>ProdStatusAll</td>
</tr>
<tr>
<td>TaxDirection</td>
</tr>
<tr>
<td>TaxType_IT</td>
</tr>
</tbody>
</table>
ENUMERATION

WHSWaveTemplateType

SQL operations made extensible

These SQL operations have been made extensible in this update.

OPERATION

InventTrans.deleteReturnTransOrigin

InventUpd_ChangeDimension.updateForceInventTrans

PriceDiscAdmCheckPost.updatePriceDiscTableRecords

ProdJournalCleanUp.deleteJournals

ProjBudgetManager.createBudgetCostForecast

ProjBudgetManager.createBudgetEmplForecast

ProjBudgetManager.createBudgetRevenueForecast

ProjBudgetManager.createBudgetSalesForecast

SubledgerJourFinalNetAmtEntryProvider.getFinalRelievingEntriesWithNetAmounts

SubledgerJourFinalRelieveEntryProvider.populateEntriesBySide

SubledgerJournalAccountEntryRelievingTmp.mergeJournalizationEntries

SubledgerJournalFinalReliever.findEntriesAlreadyRelieved

SubledgerJournalFinalReliever.findEntriesToRelieve

SubledgerJournalizer.loadFinalizeSubledgerJournalTmpDetail

SubledgerJournalizer.loadInterCompanyNonOffsetSubledgerJournalTmpDetail

SubledgerJournalizer.loadInterCompanyOffsetSubledgerJournalTmpDetail

SubledgerJournalizer.loadRelievingSubledgerJournalTmpDetail

SubledgerJournalizer.loadReversingSubledgerJournalTmpDetail

SubledgerJournalizer.loadYearEndSubledgerJournalTmpDetail

SubledgerJournalizer.summarizeJourAccountEntryDetailForRound

Metadata changes
These metadata changes have been made in this update.

**OPERATION**

Classes/CustVendSettle/classDeclaration.field modifier

Classes/LedgerPostingGeneralJournalController/transferLines.HookableAttribute

Classes/VendPaymentJournalDPnone

EcoResProductAttributeTranslationEntity.IsPublic


Enum/RDeferralsCalculatePeriod.Country region code

Enum/RDeferralsInitRetirementDate.Country region codes

Enum/RDeferralsInitWriteStartDate.Country region codes

Enum/RDeferralsInitWriteStartDate/EnumValue

Enum/RDeferralsInterval.Country region code

Enum/RDeferralsManualCalcType.Country region codes

Enum/RDeferralsMethod.Country Region Codes

Enum/RDeferralsPostValue.Country region codes

Enum/RDeferralsStatus.Country Region Code

Enum/RDeferralsTableGroupAllBook.Country Region Codes

Enum/RDeferralsTransType.Country Region Codes

Extended Data Types/AttributeValueFloat.No Of Decimals is Extensible

Increase Description in Project invoices/proposal lines for OnAccount and Expense line type

InventValue report now supports custom dimensions

Maps/AccountSumMap.Balance08,Balance08Cur,Balance09,Balance09Cur

Table/PurchTable.Visible = True

Table/VendUnrealizedRev/Field/ReversalDate.Allow edit.AllowEdit

Tables/DirPartyTable/Fields.AOS Authorization

Tables/VendSettlement/Relations/VendTrans.RelationshipType

Tables/WHSDocumentRoutingLine/Indexes/DocumentRoutingTablePrinterNameIdx
Refactored methods

These methods have been refactored to support extensibility.

<table>
<thead>
<tr>
<th>Refactored Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssetJournal.populateLedgerJournalTrans</td>
</tr>
<tr>
<td>AssetPost.postToGeneralLedger</td>
</tr>
<tr>
<td>AssetProposalDepreciation.run</td>
</tr>
<tr>
<td>AssetTransfer.addLedgerVoucherTransObjects</td>
</tr>
<tr>
<td>AxSalesLine.setDefaultDimension</td>
</tr>
<tr>
<td>BankChequeCopy.fillTmpChequePrintout</td>
</tr>
<tr>
<td>BankChequeLayout.updateDesign</td>
</tr>
<tr>
<td>BankDepositSlip.modifyBankAccountTrans</td>
</tr>
<tr>
<td>BankJournalHeaderEntity.ValidateField</td>
</tr>
<tr>
<td>BankPositivePayExport.sendFileToUser</td>
</tr>
<tr>
<td>BankStatementValidate.validateDate</td>
</tr>
<tr>
<td>BankStmtISOAccountStatement.deleteStatement</td>
</tr>
<tr>
<td>BOM.defaultField</td>
</tr>
<tr>
<td>BOM.validateField</td>
</tr>
<tr>
<td>BOM.validateWrite</td>
</tr>
<tr>
<td>BomCalcItem.insertBOMCalcTable</td>
</tr>
<tr>
<td>BomCalcItem.insertBOMCalcTrans</td>
</tr>
<tr>
<td>BomCalcProd.calcCostSheet</td>
</tr>
<tr>
<td>BOMReportFinishMax.retrieveInventTable</td>
</tr>
<tr>
<td>BudgetCaculateBalance.getActualLedgerAmountsQuery</td>
</tr>
<tr>
<td>BudgetCaculateBalance.getOriginalBudgetQuery</td>
</tr>
</tbody>
</table>
REFACTORED METHODS

BudgetCalculateBalance.getRevisedBudgetQuery

BudgetSourceCollectionIntegrator.newBudgetSourceCollectionIntegrator

BudgetSourceIntegrator.newBudgetSourceIntegrator

ChequeController.ClassDeclaration

ChequeController.init

ContactPersonApplicationSuiteEventHandlers-initializedFromCommonEventHandler

CustBillOfExchangePostRemit.postSettlingStep

CustDirectDebitMandate.checkBankIBAN

CustDueReportDetailDP.updateCustDueReportDetailTmp

CustPostInvoiceJob.custPostInvoiceUpdate

CustSettleJournalizingEntries.createGeneratedEntries

CustSettleJournalizingEntries.getInterestOriginatingEntries

CustSettleJournalizingEntries.getOriginatingEntries

CustTransDetails.new

CustTransListDP.insertCustTransListTmp

CustTransOpenCashFlow.generateCashFlow

CustVendCheque.output

CustVendCreatePaymJournal_Vend.shouldAddCustVendTransOpen

CustVendEditTaxBranchHelper_TH.init

CustVendExchAdjTrans.initLedgerVoucher

CustVendSettle.reverseTax

CustVendTransReorg.paymentSchedSplit

CustVendTransReorg.post

CustVendTransReorg.reorganize

CustVendVoucher TRANSACTION_TXT
REFACTORED METHODS

CustWriteOff.createTaxJournalLines

DimDerJourRuleProjectTimesheetsExt.getDefaultDimensionAllocation

DirPartyRolesExternallyMaintained.setFieldRestrictions

EcoResDocumentAttachmentEntity.insertDatasourceDocuRef

EcoResProductNumberBuilderVariant.getEnumeratorForEnabledOrderedProductDimensions

EFDocMsgFormat_XmlSubmit_BR.createXmlDocumentFromEFDocument

EFDocumentXpath_BR.Not applied

ERclasses - class signature

FiscalDocParmDataCreatorInvTransfer_BR.initHeaderParmData

FiscalDocParmDataCreatorInvTransfer_BR.setInventTransferTableFiscalInfo

FiscalDocumentParmDataCreator_BR.initTaxTransParmDataFromTaxTrans

FiscalDocumentParmDataCreator_BR.setAccountingAmountOnFiscalDocumentLines

FiscalDocumentPost_BR.initFiscalDocument

FreeTextInvoiceDP.insertIntoFreeTextInvoiceHeaderFooterTmp

HcmWorkerTransition.createHcmEmployment

HrpExpireWorkerLimits.expireLimitRequest,expireApprovedLimit,getAuthorityBasis

InventMov_Sales.accountBalanceSheet

InventReleaseOrderPickingForm_Sales.bldInventReleaseOrderPickingTmp

InventTestAssociationTable.checkExecutionTime

InventTrans.insertReturnTransOrigin

InventTrans.updateMarkReqTransCov

InventTransferOrderCopying_BR.createTransferLines

InventTransferOrderCopying_BR.createTransferOrder

InventTransferUpdShip.updateInventTransferLine

InventUnusedDimCleanup.isCandidateInventDimIdTable
<table>
<thead>
<tr>
<th>REFACTORED METHODS</th>
</tr>
</thead>
<tbody>
<tr>
<td>InventUpd_ChangeDimension.updateTransSwitchDim</td>
</tr>
<tr>
<td>InventUpd_ChildReference.updateLessReceipt</td>
</tr>
<tr>
<td>InventUpd_ChildReference.UpdateMoreReceipt</td>
</tr>
<tr>
<td>InventUpd_Financial.updateFinancialIssue</td>
</tr>
<tr>
<td>InventUpd_Financial.updateStdCostPrice</td>
</tr>
<tr>
<td>InventUpd_Picked.updatePickMore</td>
</tr>
<tr>
<td>InventUpd_Registered.updateRegisterLess</td>
</tr>
<tr>
<td>InventUpd_Registered.updateRegisterMore</td>
</tr>
<tr>
<td>InventUpd_Reservation.updateReserveLess</td>
</tr>
<tr>
<td>InventUpdate.initializeInventTransToIssueListFromDatabase</td>
</tr>
<tr>
<td>InventUpdate.initInventTransToReceiveList</td>
</tr>
<tr>
<td>JmgJobBundleProdFeedbackForm.getTmpJobBundleProdFeedback</td>
</tr>
<tr>
<td>JmgPaySpecificationDPClass declaration</td>
</tr>
<tr>
<td>JmgPostStandardSystem.getProjTransCostPrice</td>
</tr>
<tr>
<td>JmgPostStandardSystem.postIPCTime</td>
</tr>
<tr>
<td>JmgPostStandardSystem.postProjTime</td>
</tr>
<tr>
<td>JmgProfiles.bundleSlizeTime</td>
</tr>
<tr>
<td>JmgProfiles.insertTimeGapsPlannedAbs</td>
</tr>
<tr>
<td>JmgProfiles.sumPayEventsSec</td>
</tr>
<tr>
<td>JmgStampJournalTransfer.insertStampTrans</td>
</tr>
<tr>
<td>JmgTransferEvents.createPayEventsArray</td>
</tr>
<tr>
<td>JmgTransferEvents.insertEvents</td>
</tr>
<tr>
<td>JournalizingDefinitionManagerPurch.getDefaultJournalizingDefinition</td>
</tr>
<tr>
<td>LedgerAccrualTrans.post</td>
</tr>
<tr>
<td>LedgerAllocationBasisRules.getBasisAmount</td>
</tr>
<tr>
<td>REFACTORED METHODS</td>
</tr>
<tr>
<td>------------------------------------</td>
</tr>
<tr>
<td>LedgerJournalCheckPost.checkJournal</td>
</tr>
<tr>
<td>LedgerJournalCheckPost.postJournal</td>
</tr>
<tr>
<td>LedgerJournalCheckPost.postTransV2</td>
</tr>
<tr>
<td>LedgerJournalCheckPost.updateInterCompanyJournal</td>
</tr>
<tr>
<td>LedgerJournalTrans.markedForSettlementError</td>
</tr>
<tr>
<td>LedgerJournalTrans.validateWrite_Server</td>
</tr>
<tr>
<td>LedgerJournalTransProject.checkProjId</td>
</tr>
<tr>
<td>LedgerJournalTransUpdate.updateInterCompany</td>
</tr>
<tr>
<td>LedgerJournalTransUpdateBank.updateNow</td>
</tr>
<tr>
<td>LedgerJournalTransUpdateCust.updateNow</td>
</tr>
<tr>
<td>LedgerJournalTransUpdateLedger.createTaxLinkForTaxTransfer</td>
</tr>
<tr>
<td>LedgerJournalTransUpdateLedger.updateNow</td>
</tr>
<tr>
<td>LedgerJournalTransUpdateVend.updateNow</td>
</tr>
<tr>
<td>LedgerTransPerJournalDP.insertForLedgerBase</td>
</tr>
<tr>
<td>LedgerTransPerJournalDP.insertVoucherDetails</td>
</tr>
<tr>
<td>LedgerTransPerJournalDP.processReport</td>
</tr>
<tr>
<td>LedgerVoucher.check</td>
</tr>
<tr>
<td>LedgerVoucherGroup.end</td>
</tr>
<tr>
<td>LedgerVoucherObject.addBalanceAdjustments</td>
</tr>
<tr>
<td>LedgerVoucherObject.allocateTransaction</td>
</tr>
<tr>
<td>LedgerVoucherObject.post</td>
</tr>
<tr>
<td>LedgerVoucherObject.updateBalances</td>
</tr>
<tr>
<td>LedgerVoucherTransObject.check</td>
</tr>
<tr>
<td>Markup.copy</td>
</tr>
<tr>
<td>McrPriceHistoryLine_Sales.initAndInsertRebate</td>
</tr>
</tbody>
</table>
**REFACTORED METHODS**

Method signature: Adding contextual information before doing Unit of Measure calculation

- PdsRebateFindAndCreate.findPdsRebateAgreementLineAndCreate
- PdsRebateFindAndCreate.tamFindBillBackAgreementAndCreateClaim
- PriceDisc.findDisc
- PriceDisc.findDiscAgreement
- PriceDisc.findItemPrice
- PriceDisc.findPrice
- PriceDisc.findPriceAgreement
- PriceDiscAdmCheckPost.runFromContract
- ProdJournalCheckPost.postProdJournalTableBOM
- ProdJournalCheckPostProd.postTransLedger
- ProdJournalCreateBom.createSingleLineProdBOM
- ProdTableCleanUp.deleteProductions
- ProdUpdCostEstimation.costEstimatItems
- ProdUpdCostestimation.updateSubProdTable
- ProdUpdReportFinished.updateBomConsumption
- ProdUpdStartup.UpdateBomConsumption
- ProjAdjustment.getNewTotalCostAmount
- ProjAdjustment.setHourCostPrice
- ProjAdjustmentSplit.createNewTrans
- ProjAdjustmentSplit.initializeTmpProjAdjustmentCreate
- ProjAdjustmentUpdate_Post.post
- ProjAdjustmentUpdate_Post.postCost
- ProjAdjustmentUpdate_Post.postItem
- ProjBudget.queryProjBudgetLineCost / ProjBudgetLineCost(DataSource)
### REFACTORED METHODS

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProjBudget.queryProjBudgetLineCost / ProjBudgetLineRevenue(DataSource)</td>
<td></td>
</tr>
<tr>
<td>ProjBudgetManager.createBudgetFromForecastModel</td>
<td></td>
</tr>
<tr>
<td>ProjCategoryLookup.buildQueryTsTimesheetLine</td>
<td></td>
</tr>
<tr>
<td>ProjInvoiceChoose.main</td>
<td></td>
</tr>
<tr>
<td>ProjInvoiceJournalCreate.initJournalHeader</td>
<td></td>
</tr>
<tr>
<td>ProjInvoiceJournalPost.createProjInvoiceSalesLine</td>
<td></td>
</tr>
<tr>
<td>ProjInvoiceProposalInsertLines.doCost, doEmpl, doItem, doOnAccount, doRevenue, doSalesLine</td>
<td></td>
</tr>
<tr>
<td>ProjInvoiceProposalPeriodic.Validate</td>
<td></td>
</tr>
<tr>
<td>ProjJournalTrans.setHourCostPrice</td>
<td></td>
</tr>
<tr>
<td>ProjJournalTrans.setHourPrices</td>
<td></td>
</tr>
<tr>
<td>ProjJournalTrans.setHourSalesPrice</td>
<td></td>
</tr>
<tr>
<td>ProjPlanVersionsManager.CopyHierarchy</td>
<td></td>
</tr>
<tr>
<td>ProjPost.postCost</td>
<td></td>
</tr>
<tr>
<td>ProjPostCostProposalSale.projTransUpdate</td>
<td></td>
</tr>
<tr>
<td>ProjPostEmplProposalSale.projTransUpdate</td>
<td></td>
</tr>
<tr>
<td>ProjPostItemProposalSale.projTransUpdate</td>
<td></td>
</tr>
<tr>
<td>ProjPostRevenueProposalSale.projTransUpdate</td>
<td></td>
</tr>
<tr>
<td>ProjTable.createSalesTable_ItemReq</td>
<td></td>
</tr>
<tr>
<td>ProjTable.editSubProjects</td>
<td></td>
</tr>
<tr>
<td>psaContractLineInvoiceDP.insertTmpPSAContractLineInvoice</td>
<td></td>
</tr>
<tr>
<td>PsaGenerateQuotationLines.createSalesQuotationLines</td>
<td></td>
</tr>
<tr>
<td>PsaManageInvoiceDP.insertTmpPSAManageInvoice</td>
<td></td>
</tr>
<tr>
<td>PSAProjInvoiceDP.processLinesFromInvoiceJournal</td>
<td></td>
</tr>
<tr>
<td>PSAProjInvoiceTaxTmp.insertPSAProjInvoiceTmpForTax</td>
<td></td>
</tr>
<tr>
<td>PSAProjPostEmpIndirectProposal.indirectCreditAccountTurnover</td>
<td></td>
</tr>
</tbody>
</table>
REFACTORED METHODS

PurchCreateFromSalesOrder.autoCreatePurchOrder
PurchCreateFromSalesOrder.checkLine
PurchCreateFromSalesOrder.main
PurchCreateFromSalesOrder.querySalesLine
PurchCreateFromSalesOrder.run
PurchFormletterParmDataInvoice.copyMarkupFromPurchOrder
PurchFormletterParmDataInvoice.createInvoiceHeaderFromTempTable
PurchFormletterParmDataInvoice.createLineAsset
PurchFormletterParmDataInvoice.selectChooseLines
PurchInvoiceJournalPost.postInventory
PurchLineType.initReleasedProductSpecificDefaulting
PurchOrderLineSourceDocumentLineItem.initMonetaryAmountValue
PurchReApprovalPolicyRule.evaluatePolicy
PurchRFQAcceptJournalPost.updateSourceTable
PurchRFQSsendJournalCreate.createOrUpdateRFQLine
PurchTableForm.main
rDeferralsJournal.createTrans
rDeferralsProposalReceipt.createJournalLines
rDeferralsProposalRetirement.createJournalLines
rDeferralsProposalWritingOff.createJournalLines
rDeferralsTableMethodIterator.new
ReqDemPlanForecastAggregator.deaggregate
ReqDemPlanImportForecastService.insertDemandForecast
ReqIntercompanyDemand.initReqTransFromIntercompanyReqPO
ReqPO.Update
<table>
<thead>
<tr>
<th>REFACTORED METHODS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReqTrans.updateBOMQty</td>
</tr>
<tr>
<td>ReqTransPoMarkSumUp.updateSumUp</td>
</tr>
<tr>
<td>RequisitionReleaseStrategy.runAutoPurchOrderGeneration</td>
</tr>
<tr>
<td>RetailTransactionSalesTransMark.findInventDimFromWorkingTable; and more please review attachment</td>
</tr>
<tr>
<td>RetailTransactionSalesTransMark.updateTransactionSalesLine_InventDimId</td>
</tr>
<tr>
<td>RetailTransactionServiceOrder.createOrUpdateRetailOrderLines</td>
</tr>
<tr>
<td>RetailTransactionServiceOrders.cancelCustomerOrder</td>
</tr>
<tr>
<td>RunBaseMultiParm.initFromForm</td>
</tr>
<tr>
<td>SalesCopying.callerSalesTable.returnItem</td>
</tr>
<tr>
<td>SalesFormletterParmDataInvoice.createBasedOnPackingSlip</td>
</tr>
<tr>
<td>SalesInvoiceController.initFormLetterReport</td>
</tr>
<tr>
<td>SalesInvoiceController.parmRunOnBlockMode_TH</td>
</tr>
<tr>
<td>SalesInvoiceControllerPrintMgmtPrintSettingDetail</td>
</tr>
<tr>
<td>SalesInvoiceDP.addProductDimensionsToInventDim; tmpTaxWorkTrans;</td>
</tr>
<tr>
<td>SalesInvoiceDP.initInventDimData</td>
</tr>
<tr>
<td>SalesInvoiceDP.populateSalesInvoiceHeaderFooterTmp</td>
</tr>
<tr>
<td>SalesInvoiceDP.printBackorders</td>
</tr>
<tr>
<td>SalesInvoiceDPBase.createData</td>
</tr>
<tr>
<td>SalesInvoiceDPBase.init</td>
</tr>
<tr>
<td>SalesInvoiceJournalPost.postCustVend</td>
</tr>
<tr>
<td>SalesInvoiceJournalPost.postLine</td>
</tr>
<tr>
<td>SalesLineType.pmfValidateBatchId</td>
</tr>
<tr>
<td>SalesLineType_ReturnItem.validateField</td>
</tr>
<tr>
<td>SalesPackingSlipController.initFormLetterReport</td>
</tr>
<tr>
<td>SalesPackingSlipController.parmRunOnBlockMode_TH</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>SalesPackingSlipDP.checkPrintLineHeader</td>
</tr>
<tr>
<td>SalesPackingSlipDP.initializeInventDimReportSetup</td>
</tr>
<tr>
<td>SalesPackingSlipJournalPost.updateInventory</td>
</tr>
<tr>
<td>SalesParmTable.createPaymentSched</td>
</tr>
<tr>
<td>SalesPurchOperationTypeController_BR.getReferenceLookup</td>
</tr>
<tr>
<td>SalesQuotationDP.initializeInventDimReport</td>
</tr>
<tr>
<td>SalesQuotationProjLinkWizard.endUpdate</td>
</tr>
<tr>
<td>SalesQuotationTableType.disableFieldsIfCustomerAccountNotSpecified</td>
</tr>
<tr>
<td>SingleReturn.POSDeveloperSupport</td>
</tr>
<tr>
<td>SubledgerJournalizer.addRelievingAccountingDistributions</td>
</tr>
<tr>
<td>SubledgerJournalizer.fillPreviewTmpSummaryWithRounding</td>
</tr>
<tr>
<td>SubledgerJournalizer.loadaccountingDistributionTmp</td>
</tr>
<tr>
<td>SubledgerJournalizer.loadFinalizeSubledgerJournalTmpDetail</td>
</tr>
<tr>
<td>SubledgerJournalizer.loadInterCompanyNonOffsetSubledgerJournalTmpDetail</td>
</tr>
<tr>
<td>SubledgerJournalizer.loadInterCompanyOffsetSubledgerJournalTmpDetail</td>
</tr>
<tr>
<td>SubledgerJournalizer.loadIntercompanySubledgerJournalTmpDetail</td>
</tr>
<tr>
<td>SubledgerJournalizer.loadRelievingSubledgerJournalTmpDetail</td>
</tr>
<tr>
<td>SubledgerJournalizer.loadReversingSubledgerJournalTmpDetail</td>
</tr>
<tr>
<td>SubledgerJournalizer.loadStandardSubledgerLedgerJournalTmpDetail</td>
</tr>
<tr>
<td>SubledgerJournalizer.loadYearEndSubledgerJournalTmpDetail</td>
</tr>
<tr>
<td>SubledgerJournalizer.previewSummarizeJournalAccEntryDetail</td>
</tr>
<tr>
<td>SubledgerJournalizer.recordSubledgerJourAccEntriesForRounding</td>
</tr>
<tr>
<td>SubledgerJournalizer.recordSubledgerJournalAccountEntries</td>
</tr>
<tr>
<td>SubledgerJournalizer.summarizeJournalAccountEntryDetail</td>
</tr>
<tr>
<td>TAMTradePromotion.ValidateFundCostLevel</td>
</tr>
</tbody>
</table>
REFACTORED METHODS

taxBooksection.checkNumberSequenceSetup

TaxCalculationAdjustment.adjustBaseForTaxIncluded

TaxJournalSpec.parmTaxSpec

TaxProjInvoice.new

TaxReport770TransHandler.IT.transferTaxWithholdTrans

TaxReport770Validate_IT.validateVendors

TaxSplitPaymentPost_IT.createReverseTaxTrans

TaxWithhold.postTaxWithhold

TaxWithholdSpecialDP.createTaxWithholdSpecialTmp

TmpProjAdjustmentCreate.createFromAdjustment

TmpProjAdjustmentCreate.fieldModifiedProjId

TmpProjAdjustmentCreate.setDimension

TmpProjAdjustmentCreate.setHourCostPrice

TransactionReversal_Asset.reversalBook

TransactionReversal_Cust.createAuxiliaryCustTrans

TransactionReversal_Ledger.createGeneralJournal

TrvExpTrans.setTaxGroup

TSTimesheetEntry.setFocusOnDayIndex

TsTimesheetFavorites.createTimesheetLines

TsTimesheetFavorites.validateWrite

TSTimesheetTable.checkHours

VendAccountStatementIntDP.insertVendAccountStatementIntTmpRil

VendOutPaymControlController.Insert

VendPaymentJournalDP.insertDataFromSpecTrans

VendRequestAddVendor.init
<table>
<thead>
<tr>
<th>REFACTORED METHODS</th>
</tr>
</thead>
<tbody>
<tr>
<td>WhsContainerization.createNewContainer</td>
</tr>
<tr>
<td>WHSLoadLine.updateQtyLeftToLoad</td>
</tr>
<tr>
<td>WHSLoadTableAssignOriginInfo.assignFromFirstLoadLine</td>
</tr>
<tr>
<td>WHSLocationDirective.findPickPutLocation</td>
</tr>
<tr>
<td>WhsPostPackingSlip.alterLoadLine</td>
</tr>
<tr>
<td>WHSProdTable.pickBatchQtys</td>
</tr>
<tr>
<td>WhsWorkCreate.checkMaximums</td>
</tr>
<tr>
<td>WHSWorkCreateProdPut.insertProdParmforProdItem</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplay.buildAboveLocationDimensions</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplay.buildPick</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplay getNextFormState</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplay.processWorkLine</td>
</tr>
<tr>
<td>WHSWorkExecuteDisplayCycleCount.buildCycleCount</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplayCycleCount.buildFinish</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplayPOItemReceiving.buildPORReceiving</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplaySpotCycleCounting.displayForm</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplaySystemGrouping.displayNextForm</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplaySystemGrouping.getWorkIdFromFieldName</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplayUserDirected.displayForm</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplayUserGrouping.displayForm</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplayValidateUserDirect.displayForm</td>
</tr>
<tr>
<td>WhsWorkExecuteDisplayValidateUserDirect.validateUserDirectWorkExists</td>
</tr>
<tr>
<td>WHSWorkTable.executeWorkLinesInit</td>
</tr>
<tr>
<td>WmsJournalCheckPostReception.returnOrderUpdate</td>
</tr>
<tr>
<td>WmsPickingList_OrderPickDP.initQueryWMSOrderTrans</td>
</tr>
</tbody>
</table>
**REFACTORED METHODS**

- WmsPickingList_OrderPickDInsertTempTable
- WmsPickingList_OrderPickDOrderQty
- WmsPickingList_OrderPickDOrderUnit
- WmsPickingList_OrderPickDPrintDocumentHeader
- WmsPickingList_OrderPickDSetWMSPickingList_OrderPickTmpTemplate

**Other changes**

The following additional changes have been made for extensibility.

- Updated aging and balance list classes and forms to support the ability for customizations to increase the number of calculated aging buckets.
This is a list of extensibility features that were implemented in Dynamics 365 for Finance and Operations version 8.1.2. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

**Enumerations made extensible**

These enumerations have been made extensible in this update.

<table>
<thead>
<tr>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>DimensionHierarchyType</td>
</tr>
<tr>
<td>DirPartyType</td>
</tr>
<tr>
<td>DirPersonMaritalStatus</td>
</tr>
<tr>
<td>PrintPostCancel</td>
</tr>
<tr>
<td>INSAffiliate</td>
</tr>
<tr>
<td>LedgerJournalLinesDisplayOption</td>
</tr>
<tr>
<td>LedgerTransPerJournal</td>
</tr>
<tr>
<td>ProjDortValue</td>
</tr>
<tr>
<td>ProjPaymentStatus</td>
</tr>
<tr>
<td>RequisitionReleaseType</td>
</tr>
<tr>
<td>RetailPOSSeedDataType</td>
</tr>
<tr>
<td>SysDimension</td>
</tr>
<tr>
<td>TrvExpType</td>
</tr>
<tr>
<td>TSTimesheetEntryGridView</td>
</tr>
<tr>
<td>VendProspectiveVendorRegistrationWizardTab</td>
</tr>
</tbody>
</table>

**Metadata changes**

These metadata changes have been made in this update.
Refactored methods
These methods have been refactored to support extensibility.

AgreementConfirmationDP.getAgreementLine
AgreementConfirmationDP.getAgreementLineHistory
AssetBook.initDepreciationProfile
AssetPost.createTrueUpDepreciation
AssetPost.reduceLastDepreciation
Bank_CA.checkBankAccount
Bank_CA.checkBankRegNum
BankReconMatchingRuleAutoProcessor.doProcessMatchRule
BankReconMatchingRuleAutoProcessor.performMatchAction
REFACTORED METHODS

BomCalcItem.calcCostSheet
ChequeCopy.printCheque
ChequeDPfetch
Coupons.AddCouponTrigger
Cust.initLedgerVoucher
CustAgingReportDP.heading
CustBalanceList.constructAgingCalculation
CustCollectionLetterCreate.createJournal
CustCollectionLetterCreate.run
CustCollectionLetterPost.updateQuery
CustCollections.showAgingIndicator
CustCollectionsExcelStatement.setTransactionWorksheetHeader
CustDirectDebitMandate.lookupReference
CustDirectDebitMandate.validateMandate
CustFreeInvoiceCorrection.createAdjustingCorrectedInvoice
CustFreeInvoiceCorrection.createTaxes
CustFreeInvoiceCorrectionPost.postAdjustingInvoice
CustFreeInvoiceCorrectionPost.validate
CustInvoiceLine.insert
CustInvoicePrintJob.buildQueryForFreeText
CustInvoicePrintJob.processFreeText
CustOpenTrans.editMarkTrans
CustOpenTransReverse.markTrans
CustOverPaym.run
REFACTORED METHODS

CustPackingSlipJour.printJournal
CustPaymEntry.hasMultipleOpenTransReferences
CustPaymEntry.isInvalidOpenTransReference
CustPostInvoice.allocateNumAndVoucher
CustPostInvoice.createJournalHeader
CustRecurrenceInvoicePostService.postRecurrenceInvoice
CustSettlementPriorityProcessing.initCustTransOpen
CustStatistics.TmpStatPer.linkActive
CustTable.createRecord
CustTable.CustTable_DS/fields/CustGroup/modified
CustVendCheque.checkDataOk
CustVendCheque.output
CustVendChequeSlipTextCalculator.getMaxSlipLines
CustVendChequeSlipTextCalculator.getUnprintableReportArea
CustVendCreatePaymJournal.runPaymentProposalGenerationProcess
CustVendCreatePaymJournal.runPaymentProposalGenerationProcess
CustVendOpenTransManager.createTaxWithholding
CustVendPaymProposal.addCustVendTransOpen
CustVendReversePosting.restoreCustVendTransOpen
CustWriteOff.calcSalesTaxOnOpenTrans
CustWriteOff.generateSummarizedTmpTaxTrans
DataEntityView/ExpenseJournalLineEntity.DataEntityView/ExpenseJournalLineEntity
DirPartyPostalAddressFormHandlerExt.onUpdateTransactionCaller_delegate
Extensible class method: PriceDisc.mcrPriceDiscTableFound
FBSpedFileCreator_Contabil_BR.createRecordI052
<table>
<thead>
<tr>
<th>REFACTORED METHODS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FiscalDocumentDate_BR.lastIssueDateForSeries</td>
</tr>
<tr>
<td>HrpSigningLimitPolicyUtil.createDefaultLimit</td>
</tr>
<tr>
<td>HrpSigningLimitPolicyUtil.insertJobOrCompensationRule</td>
</tr>
<tr>
<td>HrpSigningLimitPolicyUtil.private RefRecId checkLimitAgreementDetail(HRPTmpLimitAgreementRule _tmpLimitAgreementRule, HRPAuthorityBasis _authorityBasis)</td>
</tr>
<tr>
<td>HrpWorkerLimit.private recId getAuthBaseRecId(HRPAuthorityBasis _authBasis, RefRecId _positionId)</td>
</tr>
<tr>
<td>InterCompanySyncPurchTableType.setSalesTableData</td>
</tr>
<tr>
<td>InventCountCreate_Base.doCountingBasedOnCountCode</td>
</tr>
<tr>
<td>InventMov_Purch.updateAutoLossProfit</td>
</tr>
<tr>
<td>InventMov_Purch.updateLedgerFinancial</td>
</tr>
<tr>
<td>InventMovement.addLedgerPhysicalAmounts</td>
</tr>
<tr>
<td>InventMovement.addLedgerVoucherRevenueTransactionAmountsForFinancialUpdate</td>
</tr>
<tr>
<td>InventMovement.addLedgerVoucherRevenueTransactionAmountsForPhysicalUpdate</td>
</tr>
<tr>
<td>InventMovement.addLedgerVoucherTransactionAmountsForFinancialUpdate</td>
</tr>
<tr>
<td>InventMovement.addLedgerVoucherTransactionAmountsForPhysicalUpdate</td>
</tr>
<tr>
<td>InventMovement.checkUpdatePhysical</td>
</tr>
<tr>
<td>InventMovement.processLedgerPhysicalAmountList</td>
</tr>
<tr>
<td>InventMovement.setAutoReserving</td>
</tr>
<tr>
<td>InventMovement.setCostAmountPhysical</td>
</tr>
<tr>
<td>InventMovement.updateLedgerAdjust</td>
</tr>
<tr>
<td>InventMovement.updateLedgerFinancial</td>
</tr>
<tr>
<td>InventOnhandReserve.updateReserveLot</td>
</tr>
<tr>
<td>InventUpd_Estimated</td>
</tr>
<tr>
<td>InventUpd_Estimated.updateFieldsChange</td>
</tr>
<tr>
<td>JmgPayEventsExport_Std.run</td>
</tr>
<tr>
<td>JmgStampJournalTable.approve</td>
</tr>
</tbody>
</table>
REFACTORED METHODS

JmgStampJournalTable.transfer

LedgerAccrualTrans.post

LedgerAllocationBasisRules.createGeneralJournalAccountEntrySumQuery

LedgerAllocationController.allocateAmounts

LedgerAllocationProcessRequest.allocate

LedgerJournalCheckPost.checkJournal

LedgerJournalCheckPost.postJournal

LedgerJournalDistribute.createNewJournal

LedgerJournalEngine.calculateTaxForCompleteJournal

LedgerJournalEngine.initValue

LedgerJournalTable.deleteAllLines

LedgerJournalTrans.deleteTaxUncommitted


LedgerJournalTransType.validateVoucher

LedgerJournalTransUpdate.updateIntercompany

LedgerJournalTransVendPaym./Forms/LedgerJournalTransVendPaym/Design/ActionPane(ActionPane)/ButtonGroup(ButtonGroup)/buttonCreatePayment(MenuFunctionButton)/Clicked

LedgerTransListReportHelper.buildFieldMap

LedgerTransPerJournalDP.insertForLedgerBase

LedgerVoucherObject.checkBalance

LedgerVoucherObject.checkBalanceRound

LogisticsLocationFormHandler.callerResearch

LoyaltyCardBlance.MPOS_ExtensibleViews

Macros.InventSumFields

MainAccount.DimensionAttributeValue_ds/dimensionAttributeValueIsSuspended
<table>
<thead>
<tr>
<th>REFAC'TORED METHODS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumberSeqModuleProject.loadModule</td>
</tr>
<tr>
<td>PcSourceDocumentLineUtility.initialize</td>
</tr>
<tr>
<td>PdsRebateFindAndCreate.findPdsRebateAgreementAndCreateClaim + run</td>
</tr>
<tr>
<td>PriceDisc.findPriceAgreement</td>
</tr>
<tr>
<td>PriceDisc.FindPriceAgreement.mcrPriceDiscTablefound</td>
</tr>
<tr>
<td>PriceDiscResultFields.NA</td>
</tr>
<tr>
<td>ProdJournalBOM.insertJournalCreate</td>
</tr>
<tr>
<td>ProjAdjustment.splitLine</td>
</tr>
<tr>
<td>ProjAdjustmentSplit.calculateQty</td>
</tr>
<tr>
<td>ProjAdjustmentSplit.getNewTotalSaleAmount</td>
</tr>
<tr>
<td>ProjAdjustmentUpdate.newPostAdjustment</td>
</tr>
<tr>
<td>ProjAdjustmentUpdate.run</td>
</tr>
<tr>
<td>ProjAdjustmentUpdate.transCostNew / transEmplNew / transItemNew methods</td>
</tr>
<tr>
<td>ProjAdjustmentUpdate.transItemNew</td>
</tr>
<tr>
<td>ProjAdjustmentUpdate.updateAdjusted</td>
</tr>
<tr>
<td>ProjBudgetImport.SourceType - modified</td>
</tr>
<tr>
<td>ProjBudgetRevision.updateGridHelper</td>
</tr>
<tr>
<td>ProjectPosting.getProjectLedgerDimension</td>
</tr>
<tr>
<td>ProjForecastEmpl.initValue</td>
</tr>
<tr>
<td>ProjFormletterParmData.updateQueryBuild</td>
</tr>
<tr>
<td>ProjGrant.canSubmitToWorkflow</td>
</tr>
<tr>
<td>ProjInvoiceChoose.doCost</td>
</tr>
<tr>
<td>ProjInvoiceChoose.doEmpl</td>
</tr>
<tr>
<td>ProjInvoiceChoose.doItem</td>
</tr>
<tr>
<td>ProjInvoiceChoose.doOnAccount</td>
</tr>
<tr>
<td>REFACTORED METHODS</td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td>PurchCopying.updatePriceDiscLineChangePolicy</td>
</tr>
<tr>
<td>PurchCreateFromSalesOrder.run</td>
</tr>
<tr>
<td>PurchCreateOrder.PurchTable.write</td>
</tr>
<tr>
<td>PurchEditLines.Choose_Button.click</td>
</tr>
<tr>
<td>PurchEditLines.run</td>
</tr>
<tr>
<td>PurchFormLetter.prePromptInit</td>
</tr>
<tr>
<td>PurchFormLetter.reSelect</td>
</tr>
<tr>
<td>PurchFormLetter::main</td>
</tr>
<tr>
<td>PurchFormLetterParmDataInvoice.reSelectLines</td>
</tr>
<tr>
<td>PurchInvoiceJournalCreate.allocateNumAndVoucher</td>
</tr>
<tr>
<td>PurchReqAddItem.N/A: Variable Change, not Method</td>
</tr>
<tr>
<td>PurchRFQCaseTable.isCalledFromPurchRFQCTListPageProject</td>
</tr>
<tr>
<td>PurchTable.ConvertCurrencyCode</td>
</tr>
<tr>
<td>PurchTable.create</td>
</tr>
<tr>
<td>PurchTable.create (PurchTable datasource)</td>
</tr>
<tr>
<td>PurchTableType.validateDelete</td>
</tr>
<tr>
<td>ReqCalc.actionCalcItem</td>
</tr>
<tr>
<td>ReqCalc.covCalcDim</td>
</tr>
<tr>
<td>ReqCalc.covCodeQtyMinMax</td>
</tr>
<tr>
<td>ReqCalc.covCreatePlannedOrder</td>
</tr>
<tr>
<td>ReqCalc.covCreateSafetyInvent</td>
</tr>
<tr>
<td>ReqCalc.createSafetyInvent</td>
</tr>
<tr>
<td>ReqCalc.createSafetyInventKey</td>
</tr>
<tr>
<td>ReqCalc.deleteTransactionAndCoverage</td>
</tr>
<tr>
<td>ReqCalc.setParameters</td>
</tr>
<tr>
<td>REFACTORED METHODS</td>
</tr>
<tr>
<td>-------------------------------------</td>
</tr>
<tr>
<td>ReqCalc.writeValueInventSum</td>
</tr>
<tr>
<td>ReqTransCache.listCovDimSorted</td>
</tr>
<tr>
<td>ReqTransPoMarkFirm.create</td>
</tr>
<tr>
<td>RequisitionPurchaseOrderGeneration.updateEmptyVendAccountsForManualCreation</td>
</tr>
<tr>
<td>RequisitionPurchaseOrderGeneration.validatePurchReqLine</td>
</tr>
<tr>
<td>RetailInternalOrganization.insert</td>
</tr>
<tr>
<td>RetailKitAssemblyOrder.createOrUpdateBOMJournal</td>
</tr>
<tr>
<td>RetailKitAssemblyOrder.createOrUpdateBOMJournalLine</td>
</tr>
<tr>
<td>RetailStatementPost.postRetailSpecific</td>
</tr>
<tr>
<td>RetailStoresToDeploy.setAllowEditTrue</td>
</tr>
<tr>
<td>RetailTransactionSalesTransMark.findInventDimIdFromWorkingTable</td>
</tr>
<tr>
<td>RetailTransactionSalesTransMark.populateTransactionSalesLineWorkingTable</td>
</tr>
<tr>
<td>RetailTransactionServiceOrders.cancelCustomerOrder</td>
</tr>
<tr>
<td>RetailTransactionServiceOrders.createCustomerOrder</td>
</tr>
<tr>
<td>RetailTransactionServiceOrders.createLedgerJournalTransForPayment</td>
</tr>
<tr>
<td>RetailTransactionServiceOrders.createRetailOrderPayment</td>
</tr>
<tr>
<td>RetailTransactionServiceOrders.invoiceSalesOrder</td>
</tr>
<tr>
<td>RetailTransactionServiceOrders.settleCustomerOrder</td>
</tr>
<tr>
<td>SalesCopying.canClose</td>
</tr>
<tr>
<td>SalesCreateOrder.updateDeliveryAddress</td>
</tr>
<tr>
<td>SalesFormLetter.main</td>
</tr>
<tr>
<td>SalesFormLetter.mainOnServer</td>
</tr>
<tr>
<td>SalesFormLetter.reSelect</td>
</tr>
<tr>
<td>SalesInvoiceJournalCreateBase.createJournalHeader</td>
</tr>
<tr>
<td>SalesInvoiceJournalPostBase.postLine</td>
</tr>
</tbody>
</table>
REFACTORED METHODS

SalesInvoiceJournalPostBase.updateInventory
SalesLine.createLinesFromTmpFrmVirtual
SalesLine.runPriceDiscPolicyDialog
SalesLineType_ProjectSales.canBeInvoiced
SalesPurchLine.setPriceAgreement
SalesPurchLineInterface.setPriceAgreement
SalesPurchLineInterface.setPriceDisc
SalesQuotationEditLinesForm method createParmLine
SalesQuotationListPageInteraction.linkActive
SalesQuotationProjLinkWizard.endUpdate
SalesQuotationTable.convertCurrencyCode
SalesQuotationTable.modified (SalesQuotationLine_ItemId form control)
SalesQuotationTableType.numberSeqFormHandlerQuotationId
SalesQuotationTransferToProject.createForecastOnAcc
SalesQuotationTransferToProject.createProject
SalesTable.convertCurrencyCode
SalesTable.modified
SalesTable.updateDeliveryAddress
SmaServiceFunctionLine.getFromDialog
smmBusRelTable.updateCustTable
smmBusRelTable.updateVendTable
SourceDocumentBalanceProvider.calculateEncumberedAmount
Table/MyAddressBook.xds
Table/TrvExpTrans.update
Tax.allocateInTaxWorkTrans
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TaxCalculationJournal.saveTaxTransfer</td>
<td></td>
</tr>
<tr>
<td>TaxCashDisc.calcAndInsertTaxes</td>
<td></td>
</tr>
<tr>
<td>TaxData.find</td>
<td></td>
</tr>
<tr>
<td>TaxInventTransferInvoice_BR.post</td>
<td></td>
</tr>
<tr>
<td>TaxReversePrePayment.calcPostAndInsertTaxes</td>
<td></td>
</tr>
<tr>
<td>TaxReverseTax.insertTaxWorkTrans</td>
<td></td>
</tr>
<tr>
<td>TaxReverseTax.newTrans</td>
<td></td>
</tr>
<tr>
<td>TaxSettlement.retailCalcAndInsertTaxes</td>
<td></td>
</tr>
<tr>
<td>TaxWithHold.createTaxWithholdTrans</td>
<td></td>
</tr>
<tr>
<td>TaxWithhold.postTaxWithhold</td>
<td></td>
</tr>
<tr>
<td>TransactionReversal.updateTaxTrans</td>
<td></td>
</tr>
<tr>
<td>TransactionReversal_Vend.reversal</td>
<td></td>
</tr>
<tr>
<td>TransactionTxt.setKey1</td>
<td></td>
</tr>
<tr>
<td>TransactionTxt.setKey2</td>
<td></td>
</tr>
<tr>
<td>TransactionTxt.setKey3</td>
<td></td>
</tr>
<tr>
<td>TrvExpTrans.insertPerDiemDataLines</td>
<td></td>
</tr>
<tr>
<td>TrvPbsMainDataLines.clicked</td>
<td></td>
</tr>
<tr>
<td>TrvPostExpenseHeader.postCustVendTransactions</td>
<td></td>
</tr>
<tr>
<td>TSTimesheetTrans.getCostPrice</td>
<td></td>
</tr>
<tr>
<td>VendOutPaym_Cheque.generatePaymentLines</td>
<td></td>
</tr>
<tr>
<td>VendOutPaym_RBC.generatePaymentLines</td>
<td></td>
</tr>
<tr>
<td>VendOutPaymRecord_RBC_Credit.fillField03</td>
<td></td>
</tr>
<tr>
<td>VendOutPaymRecord_RBC_Credit.fillField07</td>
<td></td>
</tr>
<tr>
<td>WhsControlItemld.populate</td>
<td></td>
</tr>
<tr>
<td>WHSCycleCountCreatePlan.insertWorkLine</td>
<td></td>
</tr>
</tbody>
</table>
The following table lists additional changes that have been made for extensibility.

- Create a SysQueryUpdateRecordSet class in AppCommon.
- Enable percent controlled for a catch weight item.
This is a list of extensibility features that were implemented in Dynamics 365 for Finance and Operations version 8.1.1. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Enumerations made extensible

These enumerations have been made extensible in this update.

<table>
<thead>
<tr>
<th>ENUMERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BankCodeType</td>
</tr>
<tr>
<td>CountryRegionType</td>
</tr>
<tr>
<td>MainAccountDimensionListProviderType</td>
</tr>
<tr>
<td>ProdSchedulingSortType</td>
</tr>
<tr>
<td>ProjAccountTypeSales</td>
</tr>
<tr>
<td>ProjBudgetBalancesGroupByOptions</td>
</tr>
<tr>
<td>ProjListStateType</td>
</tr>
<tr>
<td>ProjStatementType</td>
</tr>
<tr>
<td>SalesDeliveryDateControlType</td>
</tr>
</tbody>
</table>

Refactored methods

These methods have been refactored to support extensibility.

<table>
<thead>
<tr>
<th>REFACTORED METHODS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssetJournal</td>
</tr>
<tr>
<td>AXSalesQuotationTable.setQuotationId</td>
</tr>
<tr>
<td>BankStatementBankAccountIdentify.searchBankAccountTable</td>
</tr>
<tr>
<td>REFACTORED METHODS</td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td>BankStatementValidate.doValidate</td>
</tr>
<tr>
<td>BankStatementValidate.validateDate</td>
</tr>
<tr>
<td>BankStatementValidate.validatePeriodGap</td>
</tr>
<tr>
<td>BankStatementValidate.validatePeriodOverlap</td>
</tr>
<tr>
<td>BOM.validateWrite</td>
</tr>
<tr>
<td>BomCalcDialog.updateBomRoute</td>
</tr>
<tr>
<td>BOMCalcItem.createBomCalcItemAndAddToListBom</td>
</tr>
<tr>
<td>BOMCalcTable.transferToSalesLine</td>
</tr>
<tr>
<td>BOMCalcTable.transferToSalesQuotationLine</td>
</tr>
<tr>
<td>BomConsistOf.init</td>
</tr>
<tr>
<td>BomLevelCalc.loadDependencies</td>
</tr>
<tr>
<td>BOMReportFinishMax.init</td>
</tr>
<tr>
<td>BOMReportFinishMax.update</td>
</tr>
<tr>
<td>BOMReportFinishMax.updateBOMId</td>
</tr>
<tr>
<td>BudgetTransactionManager.checkBudgetTransactionNumberSequence</td>
</tr>
<tr>
<td>Commission.run</td>
</tr>
<tr>
<td>Cust/VendTableChangeProposalApply.apply</td>
</tr>
<tr>
<td>CustAccountStatementExtController.runPrintMgmt</td>
</tr>
<tr>
<td>CustDebitCreditNoteDPinsertForQuantity</td>
</tr>
<tr>
<td>CustDebitCreditNoteDPinsertForValue</td>
</tr>
<tr>
<td>CustInterestJour.findCustUnPostedInterestNote</td>
</tr>
<tr>
<td>CustOpenTrans.editMarkTrans</td>
</tr>
<tr>
<td>CustPackingSlipJour.PrintJounal</td>
</tr>
<tr>
<td>CustTable.openInvoiceBalanceMST</td>
</tr>
<tr>
<td>CustTable.openInvoiceBalanceMSTDoc</td>
</tr>
</tbody>
</table>
REFACTORED METHODS

CustTable.openPaymentBalanceMST
CustTable.openPaymentBalanceMSTDoc
CustTable.openPaymentBalanceMSTDue
CustVendCreatePaymJournal_Vend.searchTransactions
CustVendDisputeHelper.update
CustVendPaymProposalTransferToJournal.ClassDeclaration
CustVendPaymProposalTransferToJournal.updateSpecTransSet
CustVendPaymProposalTransferToJournal.updateSpecTransSingle
CustVendPrePaymentReversal.construct
CustVendSettle.settleForDifferentProfilesOrPrepayment
CustVendSumForPaym.run
CzCustPostAdvanceInvoice.run
DirPartyFormHandler.manageFields
EcoResProductCreate.close
EcoResProductCreate.templateRecords2Controls
EcoResProductDetailsExtended.InventTable.validateWrite
EssPersonSigningLimits/FormDataSourceRoot/HRPLimitRequestApproved.executeQuery
FormletterJournalPost.postLineDiscount
FormLetterParmData.updateQueryDocumentRanges
FormletterService.run
FormletterServiceBatchTaskManager.createFormletterParmDataTasks
FormletterServiceBatchTaskManager.createFormletterServiceTasks
FormletterServiceMultithread.newFormletterServiceMultiThread
FreeTextInvoiceController.preRunModifyContract
FreeTextInvoiceController.runPrintMgmt
### REFACTORED METHODS

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeneralLedgerExtension.validateReferenceNumber</td>
</tr>
<tr>
<td>InterCompanyPost.formLetterCollect</td>
</tr>
<tr>
<td>InterCompanyPost.formLetterCollect</td>
</tr>
<tr>
<td>InterCompanySyncPurchLineType.createOrUpdateSalesLine</td>
</tr>
<tr>
<td>InterCompanySyncPurchLineType.synchronizeInTradeCompany</td>
</tr>
<tr>
<td>InterCompanySyncSalesLineType.classDeclaration</td>
</tr>
<tr>
<td>IntrastatTransfer.updateQuery</td>
</tr>
<tr>
<td>InventBatch.insert</td>
</tr>
<tr>
<td>InventBatchConsumptionValidator.ValidateExpiryDate</td>
</tr>
<tr>
<td>InventDimCtrl_Frm_OnHand.modifyQueryBasedOnDatasourceName</td>
</tr>
<tr>
<td>InventItemBarcode.validateWrite</td>
</tr>
<tr>
<td>InventItemPrice.init</td>
</tr>
<tr>
<td>InventItemPriceSim.moveSimulatedToCurrent</td>
</tr>
<tr>
<td>InventMov_Transfer.updateLedgerFinancial</td>
</tr>
<tr>
<td>InventMovement.costValueChanged</td>
</tr>
<tr>
<td>InventMovement.updateReservation</td>
</tr>
<tr>
<td>InventOnhandReserve.ReserveLine.clicked</td>
</tr>
<tr>
<td>InventQualityManagementCreate.createPerQualityAssociations</td>
</tr>
<tr>
<td>InventQualityManagementCreate.createPerQualityAssociations</td>
</tr>
<tr>
<td>InventQualityOrderValidate.main</td>
</tr>
<tr>
<td>InventShelfLifeCriteria.initFromMovement</td>
</tr>
<tr>
<td>InventSplitTrans.check</td>
</tr>
<tr>
<td>InventTableModule.initFromInventItemPriceSim</td>
</tr>
<tr>
<td>InventTableModule.update</td>
</tr>
<tr>
<td>InventTrans.setSumAmount</td>
</tr>
<tr>
<td>REFACTORED METHODS</td>
</tr>
<tr>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>InventTrans.updateSumUp</td>
</tr>
<tr>
<td>InventTransferOrders.InventBatchId.validate</td>
</tr>
<tr>
<td>InventTransferupd.createInventTransferJourLine</td>
</tr>
<tr>
<td>InventTransferUpd.createInventTransferJourLine</td>
</tr>
<tr>
<td>InventTransPick.ctrlUpdate.clicked</td>
</tr>
<tr>
<td>InventTransPick.InventDim.InventBatchId.Validate</td>
</tr>
<tr>
<td>InventTransPick.TmpInventDim.InventBatchId.validate</td>
</tr>
<tr>
<td>InventTransRegister.InventDim.InventBatchId.validate</td>
</tr>
<tr>
<td>InventTransRegister.TmpInventDim.InventBatchId.validate</td>
</tr>
<tr>
<td>InventTransWMS_Register.updateInventFromMovementServer</td>
</tr>
<tr>
<td>InventTransWMS_Register.updateInventFromMovementServer</td>
</tr>
<tr>
<td>InventUpd_Arrived.updateArrivedMorer</td>
</tr>
<tr>
<td>InventUpd_FinancialLite.updateTrans</td>
</tr>
<tr>
<td>InventUpd_Physical.displayErrorsIfIssueQtyGreaterThanPhysical</td>
</tr>
<tr>
<td>InventUpd_Physical.updateMovementBasedOnPhysicalQty</td>
</tr>
<tr>
<td>InventUpd_Picked.updatePickLess</td>
</tr>
<tr>
<td>InventUpd_Reservation.updateReserveMore</td>
</tr>
<tr>
<td>InventUpd_WHSReservation.updateReserveMore</td>
</tr>
<tr>
<td>InventUpdate.updateDimReserveChange</td>
</tr>
<tr>
<td>InventValueReportInit.initInstrumentation</td>
</tr>
<tr>
<td>JmgJobBundle.loadActiveJobs()</td>
</tr>
<tr>
<td>JmgJobBundle.private void loadActiveJobs</td>
</tr>
<tr>
<td>JmgJobBundleProjStartupForm.getTmpJobBundleProjStartup</td>
</tr>
<tr>
<td>JmgJobBundleProjStartupForm.onClose</td>
</tr>
<tr>
<td>JmgJobBundleProjStartupForm.validateCategoryId</td>
</tr>
<tr>
<td>REFACTORED METHODS</td>
</tr>
<tr>
<td>--------------------</td>
</tr>
<tr>
<td>JmgPayAdjustment.insertAdjustment</td>
</tr>
<tr>
<td>JmgPieceRateCalc.calcPieceRate</td>
</tr>
<tr>
<td>JmgPieceRateCalc.insertEvents</td>
</tr>
<tr>
<td>JmgPostStandardSystem.createReportFinishedJournal</td>
</tr>
<tr>
<td>JmgProfileSpec.promptForAbsence</td>
</tr>
<tr>
<td>JmgStampJournalTrans.insert</td>
</tr>
<tr>
<td>JmgStampJournalTrans.update</td>
</tr>
<tr>
<td>JmgTransaction_Proj.postChange</td>
</tr>
<tr>
<td>JmgTransaction_Proj.postChange</td>
</tr>
<tr>
<td>LedgerAllocationController.allocateAmounts</td>
</tr>
<tr>
<td>LedgerAllocationRequest.closeOk</td>
</tr>
<tr>
<td>LedgerAllocationRequest.run</td>
</tr>
<tr>
<td>LedgerExchAdj.calculateAdjustments</td>
</tr>
<tr>
<td>LedgerExchAdj.constructTargetToSourceMap</td>
</tr>
<tr>
<td>LedgerJournalCheckPost.postJournal</td>
</tr>
<tr>
<td>LedgerJournalEngine.findSettledAmount</td>
</tr>
<tr>
<td>LedgerJournalTransUpdateVend.checkVoucher</td>
</tr>
<tr>
<td>LedgerParameters/FormDataSourceRoot/RDeferralsParameters.init</td>
</tr>
<tr>
<td>LedgerPostingGeneralJournalController.getLineValues</td>
</tr>
<tr>
<td>LedgerPostingGeneralJournalController.transferReferences</td>
</tr>
<tr>
<td>LedgerVoucher.check</td>
</tr>
<tr>
<td>LedgerVoucherTransObject.check</td>
</tr>
<tr>
<td>LedgerVoucherTransObject.check</td>
</tr>
<tr>
<td>MainAccount.init</td>
</tr>
<tr>
<td>MainAccount.MainAccount_ds/write</td>
</tr>
<tr>
<td>REFACTORED METHODS</td>
</tr>
<tr>
<td>--------------------</td>
</tr>
<tr>
<td>MainAccount.MainAccountLegalEntity_DS/legalEntityIsSuspended</td>
</tr>
<tr>
<td>MainAccountTemplate.rolldownChanges</td>
</tr>
<tr>
<td>Markup.resolveOrigQty</td>
</tr>
<tr>
<td>MarkupAdjustment.run</td>
</tr>
<tr>
<td>MarkupAllocation.calculateValueNow</td>
</tr>
<tr>
<td>MarkupAllocation_VendInvoiceTrans.dialog</td>
</tr>
<tr>
<td>MarkupCopy.copyFromPurchOrder</td>
</tr>
<tr>
<td>MarkupTrans.checkKeep</td>
</tr>
<tr>
<td>Method signature change</td>
</tr>
<tr>
<td>OMLegalEntity.init</td>
</tr>
<tr>
<td>OMorganizationHierarchy.updatePreviewPane</td>
</tr>
<tr>
<td>PdsBatchAttribReserve.ReserveLine.clicked</td>
</tr>
<tr>
<td>PdsBatchAttributesInput.init</td>
</tr>
<tr>
<td>PdsRebateFindAndCreate.private void calculateSums()</td>
</tr>
<tr>
<td>PdsRebateFindAndCreate.protected void createZeroRebate(PdsRebateAgreement _pdsRebateAgreement)</td>
</tr>
<tr>
<td>PdsRebateFindAndCreate.resetTransSums</td>
</tr>
<tr>
<td>PdsResetDispositionStatus.main</td>
</tr>
<tr>
<td>pdsResetShelfDates.init</td>
</tr>
<tr>
<td>PdsResetDispositionStatus.run</td>
</tr>
<tr>
<td>PdsUpdateExpDate.run</td>
</tr>
<tr>
<td>PdsUpdateShelfAdvice.run</td>
</tr>
<tr>
<td>PriceConvert_Currency.parmPrice</td>
</tr>
<tr>
<td>PriceDisc.calcPriceAmount</td>
</tr>
<tr>
<td>PriceDisc.resetPrice</td>
</tr>
<tr>
<td>PriceDiscLine.hasOnlyLineAmount</td>
</tr>
<tr>
<td>REFACTORED METHODS</td>
</tr>
<tr>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PriceDiscLine.lineAmountModified</td>
</tr>
<tr>
<td>ProdJournalCheckPostRoute.postTransLedger</td>
</tr>
<tr>
<td>ProdJournalCreateBOM.createSingleLineProdBOM</td>
</tr>
<tr>
<td>ProdUpdCostEstimation.createProdTable</td>
</tr>
<tr>
<td>ProdUpdCostEstimation.pmfCreateSubProdTable</td>
</tr>
<tr>
<td>ProdUpdReportFinished.updateBOMConsumption</td>
</tr>
<tr>
<td>ProdUpdStartUp.updateBOMConsumption</td>
</tr>
<tr>
<td>ProjBudgetTransactionManager.getTotalTransactionBudget</td>
</tr>
<tr>
<td>ProjControlPosting.queryNext</td>
</tr>
<tr>
<td>ProjFormLetter.run</td>
</tr>
<tr>
<td>ProjGroupChange.run</td>
</tr>
<tr>
<td>ProjInvoiceChooseNormal.doProposal</td>
</tr>
<tr>
<td>ProjInvoiceChooseNormal.initQuery</td>
</tr>
<tr>
<td>ProjInvoiceJournalCreate.exchRateSet</td>
</tr>
<tr>
<td>ProjInvoiceJournalPost.insertProforma</td>
</tr>
<tr>
<td>ProjInvoiceJournalPost.matchInvoicePackingSlip</td>
</tr>
<tr>
<td>ProjInvoiceJournalPost.postCustVend</td>
</tr>
<tr>
<td>ProjInvoiceProposalCreateLinesBase.doDeduction</td>
</tr>
<tr>
<td>ProjInvoiceProposalCreateLinesBase.doSalesLine</td>
</tr>
<tr>
<td>ProjInvoiceProposalInsertLines.doRevenue</td>
</tr>
<tr>
<td>ProjInvoiceSelect.queryBuild</td>
</tr>
<tr>
<td>ProjInvoiceSelect.run</td>
</tr>
<tr>
<td>ProjPost.newCreateProjTransItemCostAdjustNeg</td>
</tr>
<tr>
<td>ProjPost.postCost</td>
</tr>
<tr>
<td>ProjPostCostTransCost_Adj.projTransUpdate</td>
</tr>
<tr>
<td>REFACTORED METHODS</td>
</tr>
<tr>
<td>--------------------</td>
</tr>
<tr>
<td>ProjPostCostTransSale_Adj.projTransUpdate</td>
</tr>
<tr>
<td>ProjPostEmplJournal.projTransCreate</td>
</tr>
<tr>
<td>ProjPostEmplTransCost_Adj.projTransUpdate</td>
</tr>
<tr>
<td>ProjPostEmplTransSale_Adj.projTransUpdate</td>
</tr>
<tr>
<td>ProjPostItemTransSale_Adj.projTransUpdate</td>
</tr>
<tr>
<td>ProjPostRevenueJournal.projTransCreate</td>
</tr>
<tr>
<td>ProjProposalJour.insert</td>
</tr>
<tr>
<td>ProjSalesItemReq.clicked</td>
</tr>
<tr>
<td>ProjSalesItemReq.run</td>
</tr>
<tr>
<td>ProjStatusUpd.main</td>
</tr>
<tr>
<td>ProjTable.checkAccount</td>
</tr>
<tr>
<td>ProjTable.createSalesTable_ItemReq</td>
</tr>
<tr>
<td>ProjTable.initFromCustTable</td>
</tr>
<tr>
<td>ProjTable.insert</td>
</tr>
<tr>
<td>ProjTable.update</td>
</tr>
<tr>
<td>ProjTable.numberSeqFormHandler</td>
</tr>
<tr>
<td>ProjTable.validateWrite</td>
</tr>
<tr>
<td>ProjTable/FormDataSourceRoot/ProjTable.createFindRanges</td>
</tr>
<tr>
<td>ProjTableCreate.initValue()</td>
</tr>
<tr>
<td>ProjTableWizard.editProject</td>
</tr>
<tr>
<td>ProjTableWizardCtrl.createProject</td>
</tr>
<tr>
<td>ProjWorkBreakdownStructureV2.updateControls</td>
</tr>
<tr>
<td>PsaQuotationsController.quoteLanguageId</td>
</tr>
<tr>
<td>PurchAutoCreate method setPurchTable</td>
</tr>
<tr>
<td>PurchAutoCreate_PurchReq.create</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>PurchCreateFromSalesOrder_ChkIncluded_CheckBox_clicked</td>
</tr>
<tr>
<td>PurchCreateFromSalesOrderIncluded</td>
</tr>
<tr>
<td>PurchCreateFromSalesOrder_initFields</td>
</tr>
<tr>
<td>PurchCreateFromSalesOrder_SalesLine_ds_checkAllowCreate</td>
</tr>
<tr>
<td>PurchCreateFromSalesOrder_SalesLine_ds_included</td>
</tr>
<tr>
<td>PurchCreateFromSalesOrder_SalesLine_ds_specifyMinMaxQty</td>
</tr>
<tr>
<td>PurchCreateFromSalesOrder_SalesLine_ds_specifyPriceComponent</td>
</tr>
<tr>
<td>PurchCreateFromSalesOrder_SalesLine_ds_specifyVendAccount</td>
</tr>
<tr>
<td>PurchFinalizeServiceTask_checkAccountDate</td>
</tr>
<tr>
<td>PurchFormletterParmDataInvoice_createParmLine</td>
</tr>
<tr>
<td>PurchInvoiceJournalPost_lateMatchPackingSlip</td>
</tr>
<tr>
<td>PurchInvoiceJournalPost_postInventory</td>
</tr>
<tr>
<td>PurchInvoiceJournalPost_updateJournalTable</td>
</tr>
<tr>
<td>PurchInvoiceJournalPost_updateSourceLine</td>
</tr>
<tr>
<td>PurchInvoiceJournalPost_updateSourceLine</td>
</tr>
<tr>
<td>PurchLine_checkInvoiceConstraints</td>
</tr>
<tr>
<td>PurchLine_createFromTmpFrmVirtual</td>
</tr>
<tr>
<td>PurchLine_deleteSoft</td>
</tr>
<tr>
<td>PurchLine_deleteSoftClearValues</td>
</tr>
<tr>
<td>PurchLine_initFromSalesLine</td>
</tr>
<tr>
<td>PurchLine_itemName</td>
</tr>
<tr>
<td>PurchLineBackOrder_project</td>
</tr>
<tr>
<td>PurchLineType_statusChangeAllowed</td>
</tr>
<tr>
<td>PurchLineType_updateApprovedLine</td>
</tr>
<tr>
<td>PurchPurchOrderJournalCreate_initJournalHeader</td>
</tr>
<tr>
<td>REFACTORED METHODS</td>
</tr>
<tr>
<td>--------------------</td>
</tr>
<tr>
<td>PurchRFQLine.createPurchRFQReplyLine</td>
</tr>
<tr>
<td>PurchTable.delete</td>
</tr>
<tr>
<td>PurchTable.delete</td>
</tr>
<tr>
<td>PurchTable.getFinalDiscPriceDateDelegate</td>
</tr>
<tr>
<td>PurchTable.initFromVendTableIL</td>
</tr>
<tr>
<td>PurchTable.modifiedFieldWithUserInput</td>
</tr>
<tr>
<td>PurchYearEndProcess.processPurchOrder</td>
</tr>
<tr>
<td>ReqCalc.allowBatch</td>
</tr>
<tr>
<td>ReqCalc.checkInsertInventTransRecord</td>
</tr>
<tr>
<td>ReqCalc.covCreatePlannedOrder</td>
</tr>
<tr>
<td>ReqCalc.pmfCoCovCreatePlannedOrder</td>
</tr>
<tr>
<td>ReqCalcScheduleItemTable.createLoopMapFromQuery</td>
</tr>
<tr>
<td>ReqCalcScheduleItemTable.insertDataCompleteNetChange</td>
</tr>
<tr>
<td>ReqItemJournalUpdate.updateLines</td>
</tr>
<tr>
<td>ReqItemJournalUpdate.validate</td>
</tr>
<tr>
<td>ReqTransCache_Periodic.insertProcessItemsFromQuery</td>
</tr>
<tr>
<td>ReqTransPOCreate.insertFromReqPo</td>
</tr>
<tr>
<td>ReqTransPoMarkChangeType.updateType</td>
</tr>
<tr>
<td>ReqTransPoMarkFirm.createInventTransfer</td>
</tr>
<tr>
<td>ReqTransPoMarkFirm.createInventTransferJournal</td>
</tr>
<tr>
<td>ReqTransPoMarkFirm.createProdBOM</td>
</tr>
<tr>
<td>ReqTransPoMarkFirm.createProdTable</td>
</tr>
<tr>
<td>ReqTransPoMarkFirm.initInventTransferLine</td>
</tr>
<tr>
<td>ReqTransPoMarkSumUp.updateSumUp</td>
</tr>
<tr>
<td>ReqTransUpdate.initShelfLifeRef</td>
</tr>
</tbody>
</table>
REFACTORED METHODS

ReqTransUpdate.mustUpdateQty
SalesAutoCreate.setSalesTable
SalesCalcTax_Sales.calcTax
SalesCopying.copyFromSourceTable
SalesCopying.init
SalesCopying_CreditNote.updateInvoiceCreditCopy
SalesCreateOrderFromCustomercreate
SalesEditLines/FormDataSourceRoot/CustAdvanceInvoiceTable/Method/init
SalesFormletterParmData.createParmLine
SalesFormletterParmData.initSalesParmUpdateFormletter
SalesFormletterParmData.updateQueryBuild
SalesInvoiceController.preRunModifyContract
SalesInvoiceController.runPrintMgmt
SalesInvoiceDPBase.getMarkUpTaxCode
SalesInvoiceDPBase.initLocalizationData
SalesInvoiceJournalCreateBase.createJournalHeader
SalesInvoiceJournalPostBase.createReportData
SalesInvoiceJournalPostBase.postLine
SalesInvoiceJournalPostBase.updateJournalLine
SalesInvoiceJournalPostBase.updateJournalTable
SalesJournalSelect_Invoice.closeOK
SalesLine.returnUpdateBasedOnDispcode
SalesLineCopyFromSource.updateSalesLine
SalesLineType.formProduction
SalesLineType.initFromCustInvoiceTrans
<table>
<thead>
<tr>
<th>REFACTORED METHODS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SalesLineType.initFromSalesBasketLine</td>
</tr>
<tr>
<td>SalesLineType.initFromSalesLine</td>
</tr>
<tr>
<td>SalesLineType.InitFromSalesTable</td>
</tr>
<tr>
<td>SalesLineType.initReleasedProductSpecificDefaulting</td>
</tr>
<tr>
<td>SalesLineType.initStorage_dimensionsFromSalesTable</td>
</tr>
<tr>
<td>SalesLineType.pmfValidateBatchId</td>
</tr>
<tr>
<td>SalesLineType.setSalesStatusNonInventoried</td>
</tr>
<tr>
<td>SalesLineType.validateWrite</td>
</tr>
<tr>
<td>SalesLineType_Project.validateWrite</td>
</tr>
<tr>
<td>SalesPackingSlipJournalPost.createReportData</td>
</tr>
<tr>
<td>SalesPackingSlipJournalPost.PostInventory</td>
</tr>
<tr>
<td>SalesPackingSlipJournalPost.updateSourceLine</td>
</tr>
<tr>
<td>SalesPackingSlipJournalPostProj.writeProjTrans</td>
</tr>
<tr>
<td>SalesParmTable.createPaymentSched</td>
</tr>
<tr>
<td>SalesQuotationCopying.copyServer</td>
</tr>
<tr>
<td>SalesQuotationEditLinesForm.initializeAndRun</td>
</tr>
<tr>
<td>SalesQuotationEditLinesForm.initializeAndRun</td>
</tr>
<tr>
<td>SalesQuotationEditLinesForm_Proj_Confirm.queryBuildSalesQuotationTable</td>
</tr>
<tr>
<td>SalesQuotationEditLinesForm_Sales_Confir.numRefSalesId</td>
</tr>
<tr>
<td>SalesQuotationJumpRef.main</td>
</tr>
<tr>
<td>SalesQuotationLineCopyFromSource.updateAfterCopy</td>
</tr>
<tr>
<td>SalesQuotationLineType.salesQtyAllowEdit</td>
</tr>
<tr>
<td>SalesQuotationLineType_Proj.initFromSalesQuotationLine</td>
</tr>
<tr>
<td>SalesQuotationProjLinkWizard.endUpdate</td>
</tr>
<tr>
<td>SalesQuotationProjLinkWizard.linkQuotationToProject</td>
</tr>
<tr>
<td>REFACTORED METHODS</td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td>SalesQuotationProjLinkWizard.next</td>
</tr>
<tr>
<td>SalesQuotationTable.clicked</td>
</tr>
<tr>
<td>SalesQuotationTable.initFromBusinessRelationTable</td>
</tr>
<tr>
<td>SalesQuotationTable.initFromCustTable</td>
</tr>
<tr>
<td>SalesQuotationTable.initFromSalesQuotationTable</td>
</tr>
<tr>
<td>SalesQuotationTable.modifiedField</td>
</tr>
<tr>
<td>SalesQuotationTable.modifiedFieldDDC</td>
</tr>
<tr>
<td>SalesQuotationTable.validatewrite</td>
</tr>
<tr>
<td>SalesQuotationTable.writeCreateQuotation</td>
</tr>
<tr>
<td>SalesQuotationUpdate.main</td>
</tr>
<tr>
<td>SalesTable.clicked</td>
</tr>
<tr>
<td>SalesTable.SalesTable_ds.Create</td>
</tr>
<tr>
<td>SalesTableForm.enableUpdateJournalButtonsMultipleOrders</td>
</tr>
<tr>
<td>SalesTableType.modifiedField</td>
</tr>
<tr>
<td>SalesTableType.modifiedField</td>
</tr>
<tr>
<td>SalesTableType.validateDelete</td>
</tr>
<tr>
<td>SalesTotals.showTax</td>
</tr>
<tr>
<td>SalesTotals.showTaxLine</td>
</tr>
<tr>
<td>SalesTotals_Sales.calculateFreeValue</td>
</tr>
<tr>
<td>SubledgerJournalAccountEntryTmpSummary.getCopy</td>
</tr>
<tr>
<td>SubledgerJournalEntryBalance.initBalances</td>
</tr>
<tr>
<td>SubledgerJournalizer.validateDebitCreditBalance</td>
</tr>
<tr>
<td>SubledgerJournalizer.validateTransferEntriesBalance</td>
</tr>
<tr>
<td>SubledgerJourPennyDiffRecognizer.recognizePennyDifference</td>
</tr>
<tr>
<td>SubledgerJourSummaryRptCurRoundAdjRcgnzr.recognizeRoundingAdjustment</td>
</tr>
<tr>
<td>REFACTORED METHODS</td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td>Table/ProjTable.isCustomerTransferNeeded</td>
</tr>
<tr>
<td>Table/PurchTable.checkUpdate</td>
</tr>
<tr>
<td>Table/TrvExpTrans/Method/setDefaultProjectFromExpenseReport</td>
</tr>
<tr>
<td>TaxCalculationAdjustment.adjustBaseForAllLines</td>
</tr>
<tr>
<td>TMSMiscellaneousCharge.ValidateChargeCode</td>
</tr>
<tr>
<td>TmsProcessXML_Base.readAppPurchLine</td>
</tr>
<tr>
<td>TmsProcessXML_Base.writeShipManualAccessorials</td>
</tr>
<tr>
<td>TransactionReversal_Asset.reversalBook</td>
</tr>
<tr>
<td>TransactionReversal_Ledger.createGeneralJournal</td>
</tr>
<tr>
<td>TSTimesheetEntryQuery.initializeQuery</td>
</tr>
<tr>
<td>TsTimesheetsPost.postNoNeverLedgerTrx</td>
</tr>
<tr>
<td>VendDocumentLineType_Invoice.validateRow</td>
</tr>
<tr>
<td>VendOpenTransReverse.initFromCommon</td>
</tr>
<tr>
<td>VendorInvoiceLineSourceDocLineItem.hasMainAccDerivationInputChanged</td>
</tr>
<tr>
<td>VendPurchOrderJour.printJournal</td>
</tr>
<tr>
<td>VendReport_LedgerReconciliation.insertLedgerTransactions</td>
</tr>
<tr>
<td>VendTable.createRecord</td>
</tr>
<tr>
<td>WhsControlQty.process</td>
</tr>
<tr>
<td>WhsDocumentRouting.getRoute</td>
</tr>
<tr>
<td>WHSDocumentRouting.translate</td>
</tr>
<tr>
<td>WHSLocationDirective.validateBatchMixingOnLocation</td>
</tr>
<tr>
<td>WHSLocationDirective.validateMixingRulesAndStockingLimit</td>
</tr>
<tr>
<td>WHSPostEngineBase.prodPickQty</td>
</tr>
<tr>
<td>WHSProdTable.stopAndUnpick</td>
</tr>
<tr>
<td>WHSReverseSalesWork.createWorkToMoveItemsBack</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>WHSRFC controlData.populateData</td>
</tr>
<tr>
<td>WhsrFC controlData.processDataInternal</td>
</tr>
<tr>
<td>WHSRFC controlData.processLegacyControl</td>
</tr>
<tr>
<td>WHSSplitWork</td>
</tr>
<tr>
<td>WHSWarehouseRelease.createLoadLines</td>
</tr>
<tr>
<td>WhsWaveFormActions.printPickList</td>
</tr>
<tr>
<td>WHSWaveTable.createWaveTableFromTemplate</td>
</tr>
<tr>
<td>WhsWorkCreateProdPut.createOrUpdateBatch</td>
</tr>
<tr>
<td>WhsWorkCreateReceiving.createBatch</td>
</tr>
<tr>
<td>WHsWorkExecute.createAndPostTransferJournal</td>
</tr>
<tr>
<td>WHsWorkExecute.CreateDimTrackingRecord</td>
</tr>
<tr>
<td>WHsWorkExecuteDisplay.getNextFormState</td>
</tr>
<tr>
<td>WHsWorkExecuteDisplay.processTrackingDimDetails</td>
</tr>
<tr>
<td>WHsWorkExecuteDisplay.processTrackingDimDetails</td>
</tr>
<tr>
<td>WHsWorkExecuteDisplayAdjustIn.displayForm</td>
</tr>
<tr>
<td>WHsWorkExecuteDisplayCycleCountGrouping.getCycleCountWorkId</td>
</tr>
<tr>
<td>WHsWorkExecuteDisplayListWork.addWorkListFieldForWork</td>
</tr>
<tr>
<td>WHsWorkExecuteDisplayListWork.buildTableContents</td>
</tr>
<tr>
<td>WHsWorkExecuteDisplayListWork.getWorkQuery</td>
</tr>
<tr>
<td>WHsWorkExecuteDisplayLPReceiving.buildReceivingLPInfoFromASNItem</td>
</tr>
<tr>
<td>WHsWorkExecuteDisplayPOItemReceiving.buildPOReceiving</td>
</tr>
<tr>
<td>WHsWorkExecuteDisplayPOReceiving.buildLicensePlateLabels</td>
</tr>
<tr>
<td>WHsWorkExecuteDisplayReportAsFinishedBySerial.createPutWork</td>
</tr>
<tr>
<td>WHsWorkInventTransReservationCollectionBuilder.canMoveReservationFromWorkLine</td>
</tr>
<tr>
<td>WHsWorkUser.changePassword</td>
</tr>
</tbody>
</table>
REFACTORED METHODS

- WHSWorkUserAuthenticator.authenticate
- WmsArrivalCreateJournal.createWMSJournalTransFromArrivalDetails
- WmsJournalFormTrans.promptSplitReturnLine
- WmsJournalTransSplit.serverRun
- WMSOrder.updateReservOrderedDim
- WmsPickingRoute.finishMulti
- WrkCtrCapResHandler.new

Metadata changes

These metadata changes have been made in this update.

OPERATION

- /Data Entities/PdsItemBatchAttributeEntity.IsPublic
- /Forms/WHSMobileAppField/FormDesign/AppBar/CreateDefaultButtonGroup/CreateDefaultButton.NeededPermission
- /Forms/WMSPickingRegistration/Design/Tab(Tab)/Details(TabPage)/HeaderDetails(Tab)/PickingLinesPage(TabPage)/PickLinesGrid(Grid)/InventoryDimensionsGrid(GroupName).DataGroup
- /Table/FreeTextInvoiceLocalizationTmp.Visible
- /Tables/MarkupAutoTable/Indexes/MarkupIdx MarkupsIdx
- Data types/Extended data types/ItemVolume.NoOfDecimalsIsExtensible
- DataEntityView/EcoResProductCategoryAssignmentEntity is OData enabled
- DataEntityView/EcoResProductEntity is OData enabled
- DataEntityView/EcoResReleasedProductEntity is OData enabled
- Enum/InvoiceReferenceNumberFormulaType_FI.Country region code
- Enum/InvoiceReferenceNumberFormulaType_FI/EnumValue
- RouteOprTime.NoffDecimalsExtensible
- Table/HRPDefaultSigningLimitRuleCompensationTmp.String size
- Table/PSAProjInvoiceTmp/Properties.Title1, Title2
- Table/VendUnrealizedRev/Field/ReversalDate.Allow edit
Additional extensibility enhancements

In addition to the refactored methods, the following extensibility enhancements have been made.

- Dimension based discount
- Redesign InventPosting searching algorithms
This is a list of extensibility features that were implemented in Dynamics 365 for Finance and Operations version 8.1. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

### Refactored methods to support extensibility

These methods have been refactored to support extensibility through chain of command, delegates, or by providing access to members.

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssetPost.createTrueUpDepreciation</td>
</tr>
<tr>
<td>AssetPost.createTrueUpDepreciation</td>
</tr>
<tr>
<td>AssetPostDisposal.post</td>
</tr>
<tr>
<td>AssetPostDisposal.postVoucherTransactions</td>
</tr>
<tr>
<td>AssetPostDisposal.postVoucherTransactions</td>
</tr>
<tr>
<td>AssetTable.buildComposedOf</td>
</tr>
<tr>
<td>AssetTable.createStruct</td>
</tr>
<tr>
<td>AxInventDim_SalesLine.setInventSiteld</td>
</tr>
<tr>
<td>BankChequePrint.printDocument</td>
</tr>
<tr>
<td>BankCodaProcessing.custSettlement</td>
</tr>
<tr>
<td>BankDeposit.InitFromLedgerJournalTrans</td>
</tr>
<tr>
<td>BankExchAdj_RU.calcAndPostCurrency</td>
</tr>
<tr>
<td>BankExchAdj_RU.calcAndPostCurrency</td>
</tr>
<tr>
<td>BankExchAdj_RU.calcBalance</td>
</tr>
<tr>
<td>BankPrintTestCheque.printCheque</td>
</tr>
<tr>
<td>BankPrintTestCheque.printCheque</td>
</tr>
<tr>
<td>BankPrintTestCheque.printCheque</td>
</tr>
<tr>
<td>METHOD</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>BankPrintTestCheque.printCheque</td>
</tr>
<tr>
<td>BlindCloseView.MPOS_ExtensibleViews</td>
</tr>
<tr>
<td>BudgetAnalysisInquiryHelper_PSN.insertTreeNodes</td>
</tr>
<tr>
<td>BudgetAnalysisInquiryHelper_PSN.insertTreeNodes</td>
</tr>
<tr>
<td>BudgetAnalysisInquiryProcessor_PSN.getBudgetAnalysisLedgerDimensions</td>
</tr>
<tr>
<td>CAMStatisticalEntryTransferJournalEntryDataMapper.newFromParameters</td>
</tr>
<tr>
<td>CaseUpdateStatus_Close.changeStatus</td>
</tr>
<tr>
<td>CashManagementView.NewExtension</td>
</tr>
<tr>
<td>ChequeController.init</td>
</tr>
<tr>
<td>ChequeDP.insertChequeTmp</td>
</tr>
<tr>
<td>Class InventTransferEstimation.updateEstimatedPre</td>
</tr>
<tr>
<td>class Markup.insertReturnMarkupTrans</td>
</tr>
<tr>
<td>class MarkupTransInsert.insertForCodes</td>
</tr>
<tr>
<td>Class PurchLineType.updateSalesLine</td>
</tr>
<tr>
<td>class PurchPackingSlipJournalPost.postInventory</td>
</tr>
<tr>
<td>Class SalesQuantity_PackingSlip.calcQtyInvent</td>
</tr>
<tr>
<td>Class SalesQuantity_PackingSlip.calcQtySales</td>
</tr>
<tr>
<td>Class/AssetPost.post</td>
</tr>
<tr>
<td>Class/FormLetterJournalPost.post</td>
</tr>
<tr>
<td>Class/PurchReqTableListPageInteraction.applyFilter</td>
</tr>
<tr>
<td>Class\PurchFormLetter_PackingSlip.checkFormLetterId</td>
</tr>
<tr>
<td>Class\PurchFormletterProvider.checkLines</td>
</tr>
<tr>
<td>Class\TaxCalculationAdjustment.calcManualInserted</td>
</tr>
<tr>
<td>CompanyInfoHelper.onValidateField</td>
</tr>
<tr>
<td>CostSheetDesigner.DataSource:CostSheetCalculationFactor.validateWrite</td>
</tr>
<tr>
<td>METHOD</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>CustAccountStatementExtController.initAgingBalances</td>
</tr>
<tr>
<td>CustAccountStatementExtController.preprocessParty</td>
</tr>
<tr>
<td>CustAccountStatementExtController.processParty</td>
</tr>
<tr>
<td>CustAccountStatementExtController.processTrans</td>
</tr>
<tr>
<td>CustAccountStatementExtController.runPrintMgmt</td>
</tr>
<tr>
<td>CustCollectionLetterCreate.run</td>
</tr>
<tr>
<td>CustCollectionLetterPost.RUN</td>
</tr>
<tr>
<td>CustInterestAdjust.createCustInvoiceTable</td>
</tr>
<tr>
<td>CustInterestCreate.createJournal</td>
</tr>
<tr>
<td>CustInterestCreate.logDateRecordError</td>
</tr>
<tr>
<td>CustInterestCreate.newAccount</td>
</tr>
<tr>
<td>CustInterestCreate.resetCustInterest</td>
</tr>
<tr>
<td>CustInterestCreate.runOnce</td>
</tr>
<tr>
<td>CustInvoiceCalcTax_Invoice.updateTaxWriteCode</td>
</tr>
<tr>
<td>CustInvoiceLine.Insert</td>
</tr>
<tr>
<td>CustOutPaym.generatePaymentLines</td>
</tr>
<tr>
<td>CustQuotationJour.printJournal</td>
</tr>
<tr>
<td>CustTable.CanSubmitToWorkflow</td>
</tr>
<tr>
<td>CustTrans.documentDateModified</td>
</tr>
<tr>
<td>CustTrans.documentDateModified</td>
</tr>
<tr>
<td>CustTransSettlement.insertSettlementLines</td>
</tr>
<tr>
<td>CustVendCheque.createBankChequePaymentTrans</td>
</tr>
<tr>
<td>CustVendCheque.initTmpChequePrintout</td>
</tr>
<tr>
<td>CustVendCheque.output</td>
</tr>
<tr>
<td>CustVendChequeSlipTextCalculator.seebelow</td>
</tr>
</tbody>
</table>
CustVendCreatePaymJournal_Vend.shouldAddCustVendTransOpen
CustVendPaymProposalTransferToJournal.transferProposal
CustVendPaymSched.construct
CustVendPaymSched.initCalcPaymSched
CustVendReversePosting.restoreCustVendTransOpen
CustVendVoucher.post
DataEntityView/SalesOrderLineEntity/Method/validateWrite
DataEntityView/SalesQuotationLineEntity/Method/validateWrite
EFDocEmailProcessor_BR.saveReceivedXmlData
EFDocMsgFormat_XmlBase_BR.createElementWithValue
Extract into method: SalesParmTable.GetPaymentSched
Form InventJournalAsset\Methods\init
Form InventJournalCount\Methods\init
Form InventJournalMovement.init
Form TrvExpenses.confirmCategoryChange
Form\SalesEditLines.closeOk
FormletterJournalPost.newPostPurch
FreeTextInvoiceDP.Variable declaration
HcmActionTypeSetup.lookupWorkflowTable
HcmPosition_HcmPositionDefaultDimension_formDatasource.selectionChanged
HcmPositionTransition:: createHcmPositionWorkerAssignment
HcmPositionWorkerAssignmentDialog:: createWorkerActionOk
HRMCompFixedPlanTable::enforcePayRateTolerance
InterCompanyPostPurch.construct
InterCompanyPostPurch.formLetterUpdate
METHOD

InterCompanyPostSales.formLetterUpdate

InterCompanySyncPurchTableType.setSalesTableData

IntrastatCheck.Run

IntrastatTransfer.calcAmountsAndMarkups

IntrastatTransfer.calcValuesSign

IntrastatTransfer.distributeIntrastatAddValueLV

IntrastatTransfer.run

IntrastatTransferIT.calcCounty

IntrastatTransferIT.updateTransactionCurrencyAmount

InventAdj_Transact.run

InventCostPost.postInventCostTransVariance

InventMov_Purch.updateAutoLossProfit

InventMov_Purch.updateBuffer

InventMovement.checkUpdatePhysical

InventoryLookupMatrixView.ExtensibleViews

InventTable.pdsValidateBestBeforeDays

InventTransWMS_Register.updateInventFromMovementServer

InventUpd_CHILDReference.updateLessIssue

InventUpd_CHILDReference.updateLessReceipt

InventUpd_CHILDReference.updateMoreIssue

InventUpd_CHILDReference.updateMoreReceipt

InventUpd_Financial.newSalesInvoice

InventUpd_Physical.updatePhysicalIssue

InventUpd_Physical.updateTransPhysicalReturnedReceipt

InventUpd_Physical::newSalesPackingSlip
<table>
<thead>
<tr>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>InventUpd_Reservation.updateNow</td>
</tr>
<tr>
<td>InventUpd_Reservation.updateReserveMore</td>
</tr>
<tr>
<td>InventUpdate.updateInventTransPosting</td>
</tr>
<tr>
<td>InventUpdate.writeInventTrans</td>
</tr>
<tr>
<td>InventUpdate.writeInventTrans</td>
</tr>
<tr>
<td>InventUpdateWriteInventTrans</td>
</tr>
<tr>
<td>InventUpdateMarking.addMarking</td>
</tr>
<tr>
<td>InventUpdateMarking.removeMarking</td>
</tr>
<tr>
<td>InventUpdateReserveMore.createQueryRuns</td>
</tr>
<tr>
<td>JmgCalcApprovePickDialog.closeOk</td>
</tr>
<tr>
<td>JmgJobBundle.postTime</td>
</tr>
<tr>
<td>JmgJobBundleProdFeedbackForm.getTmpJobBundleProdFeedback</td>
</tr>
<tr>
<td>JournalStaticDataModel.initializeJournalTableFields</td>
</tr>
<tr>
<td>Ledger.populateTmpTable</td>
</tr>
<tr>
<td>Ledger.populateTmpTable</td>
</tr>
<tr>
<td>LedgerAccrualTrans_Calendar.allocate</td>
</tr>
<tr>
<td>LedgerAccrualTrans_Calendar.allocate</td>
</tr>
<tr>
<td>LedgerAccrualTrans_Fiscal.allocate</td>
</tr>
<tr>
<td>LedgerAccrualTrans_Fiscal.allocate</td>
</tr>
<tr>
<td>LedgerAutomaticTransactionAccountEntity</td>
</tr>
<tr>
<td>LedgerCreatePeriodBalances.createPeriodBalancesMainAccount</td>
</tr>
<tr>
<td>LedgerExchAdj.postAdjustment</td>
</tr>
<tr>
<td>LedgerExchAdj.run</td>
</tr>
<tr>
<td>LedgerFiscalJournalDP_IT.getDisplaySequenceNumber</td>
</tr>
<tr>
<td>LedgerJournalCheckPost.checkJournal</td>
</tr>
<tr>
<td>LedgerJournalCheckPost.postJournal</td>
</tr>
<tr>
<td>METHOD</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>LedgerJournalCheckPost.runInternal</td>
</tr>
<tr>
<td>LedgerJournalEngine.accountModified</td>
</tr>
<tr>
<td>LedgerJournalEngine.findSettledAmount</td>
</tr>
<tr>
<td>LedgerJournalEngine.formSettlement</td>
</tr>
<tr>
<td>LedgerJournalEngine.newVoucher</td>
</tr>
<tr>
<td>LedgerJournalPeriodicCopy.journalTransCopy</td>
</tr>
<tr>
<td>LedgerJournalTrans.checkVATTransaction</td>
</tr>
<tr>
<td>LedgerJournalTrans.checkVatTransaction</td>
</tr>
<tr>
<td>LedgerJournalTrans.validateWrite_Server</td>
</tr>
<tr>
<td>LedgerJournalTrans_Project.checkCategoryId</td>
</tr>
<tr>
<td>LedgerJournalTransUpdateAsset</td>
</tr>
<tr>
<td>LedgerJournalTransUpdateLedger.updateNow</td>
</tr>
<tr>
<td>LedgerJournalTransVoucherTemplates.createVoucherTemplate</td>
</tr>
<tr>
<td>LedgerJournalTransVoucherTemplates.createVoucherTemplate</td>
</tr>
<tr>
<td>LedgerJournalTransVoucherTemplates.saveVoucherTemplate</td>
</tr>
<tr>
<td>LedgerJournalTransVoucherTemplates.saveVoucherTemplate</td>
</tr>
<tr>
<td>LedgerJournalTransVoucherTemplates.SaveVoucherTemplate, CreateVoucherTemplate</td>
</tr>
<tr>
<td>LedgerPostingGeneralJournalController.addLine</td>
</tr>
<tr>
<td>LedgerPostingGeneralJournalController.getLineValues</td>
</tr>
<tr>
<td>LedgerTransferOpening.getMainAccountStorageSegment</td>
</tr>
<tr>
<td>LedgerTransferOpening.processNewClosingRecordsForOpening</td>
</tr>
<tr>
<td>LedgerTransferOpening.processQueryForPublicSector</td>
</tr>
<tr>
<td>LedgerTransModule.tax</td>
</tr>
<tr>
<td>LedgerVoucherObject.allocateTransaction</td>
</tr>
</tbody>
</table>
LedgerVoucherObject.updateLedgerPostingJournal

LogisticsEntityLocationFormHandler.manageControls

LogisticsPostalAddressFormEventHandler.updatePrimaryControl

MainAccount.canSubmitToWorkflow

Move variable declaration to first assignment when possible

NewElement.NewMethod

OmOperatingUnit.getDimensionViewId

Originaldocuments.findFromCustTrans

Originaldocuments.findFromGeneralJournal

PaymSchedCalc.createCustVendTransaction

PaymSchedCalc.initFromPurchTotals

PaymSchedCalc.initFromSalesTotals

POSApids.NewTrigger

PosAPids.NewTrigger

PriceDiscAdmCheckPost.checkForOverlapsAndGaps

PriceDiscAdmCheckPost.runFromContract

PriceDiscTable.buildSearchFilter

PrintableReceipt

ProjAdjustmentUpdate.newPostAdjustment

ProjAdjustmentUpdate.projTrans

ProjAdjustmentUpdate.transEmplNew

ProjAdjustmentUpdate_0Post.Post

ProjAdjustmentUpdate_0Post.update

ProjBudgetManager.createBudgetRevenueForecast, createBudgetSalesForecast

ProjBudgetManager.insertProjBudgetLine
<table>
<thead>
<tr>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProjBudgetManager.insertProjBudgetLine</td>
</tr>
<tr>
<td>ProjBudgetManager.updateProjBudgetLinesWithAmt</td>
</tr>
<tr>
<td>ProjCategory.categoryType2CategoryEmplOption</td>
</tr>
<tr>
<td>ProjCategory.categoryType2CategoryEmplOption</td>
</tr>
<tr>
<td>ProjCategory.categoryType2TransType</td>
</tr>
<tr>
<td>Projcategory.transType2CategoryType</td>
</tr>
<tr>
<td>Projcontrol.categoryType2CostType</td>
</tr>
<tr>
<td>Projcontrol.costType2CategoryType</td>
</tr>
<tr>
<td>ProjControlPosting.insertControlTrans, processSalesValue</td>
</tr>
<tr>
<td>ProjectSourceDocumentLineItemHelper.projOrigin and projTransType</td>
</tr>
<tr>
<td>ProjForecastListPageInteraction.runForcastFormBasedOnForecastUnion</td>
</tr>
<tr>
<td>ProjForecastReduce.newProjPost</td>
</tr>
<tr>
<td>ProjFormLetter method main/mainOnServer</td>
</tr>
<tr>
<td>ProjInvoiceCancel.cancelProposal</td>
</tr>
<tr>
<td>ProjInvoiceChoose.run</td>
</tr>
<tr>
<td>ProjInvoiceJournalCreate.createJournalHeader</td>
</tr>
<tr>
<td>ProjInvoiceLines.run</td>
</tr>
<tr>
<td>ProjInvoiceProposalCreateLines.loadLastValue</td>
</tr>
<tr>
<td>ProjInvoiceProposalCreateLines.runItemQuery</td>
</tr>
<tr>
<td>ProjInvoiceProposalInsertLines.run</td>
</tr>
<tr>
<td>ProjInvoiceProposalInsertLines.setProjProposalJour</td>
</tr>
<tr>
<td>ProjInvoiceProposalInsertLines.setProjProposalJour</td>
</tr>
<tr>
<td>ProjInvoiceProposalInsertLines.setProjProposalTotals</td>
</tr>
<tr>
<td>ProjInvoiceTableCreate.initializeValues</td>
</tr>
<tr>
<td>Projparameters.defaultProjCategory</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>ProjPeriodPostingLedgerCost.updateTrans</td>
</tr>
<tr>
<td>ProjPost.newCheckTrans</td>
</tr>
<tr>
<td>Projpost.newCreateProjTransAndLedger</td>
</tr>
<tr>
<td>Projpost.newCreateProjTransAndLedgerAdj</td>
</tr>
<tr>
<td>Projpost.newTransAdjNegative/journal</td>
</tr>
<tr>
<td>Projpost.postcost</td>
</tr>
<tr>
<td>ProjSalesItemReq.modified</td>
</tr>
<tr>
<td>ProjSplitBill.transType</td>
</tr>
<tr>
<td>ProjSplitBill.transType</td>
</tr>
<tr>
<td>ProjTable.generateNextSubProjectId</td>
</tr>
<tr>
<td>ProjTableCreate.canClose</td>
</tr>
<tr>
<td>ProjTableLookup.isJournal</td>
</tr>
<tr>
<td>ProjTableWizardCtrl.createProject</td>
</tr>
<tr>
<td>PsaProjAndContractInvoiceController.getReportTitle</td>
</tr>
<tr>
<td>PsaProjAndContractInvoiceController.getReportTitle</td>
</tr>
<tr>
<td>PsaProjAndContractInvoiceController.getReportTitle</td>
</tr>
<tr>
<td>PSAProjRetainerInvoicing.run</td>
</tr>
<tr>
<td>PsaQuotationsController.getReportTitle</td>
</tr>
<tr>
<td>PurchaseOrderResponseConsume.consumeChangesToPurchLineAndUpdateResponelineConsumptionState</td>
</tr>
<tr>
<td>PurchaseOrderResponseConsume.consumeChangesToPurchLineAndUpdateResponelineConsumptionState</td>
</tr>
<tr>
<td>PurchaseOrderResponseConsume.consumeChangesToPurchLineAndUpdateResponelineConsumptionState</td>
</tr>
<tr>
<td>PurchAutoCreate_Sales.createPurchTable</td>
</tr>
<tr>
<td>PurchAutoCreate_Sales.setPurchTable</td>
</tr>
<tr>
<td>PurchCalcTax_Invoice.updateTaxWriteCode</td>
</tr>
<tr>
<td>PurchCopying.copyLine</td>
</tr>
</tbody>
</table>
METHOD

PurchCreateFromSalesOrder.autoCreatePurchOrder

PurchFormletter_Invoice.newInvoice

PurchFormletter_Packingslip.runProjectPostings

PurchFormletterParmData.createParmLine

PurchFormletterParmData.newData

PurchFormLetterParmData.reSelectLines

PurchFormletterParmDataInvoice.chooseLinesNext

PurchFormletterParmDataInvoice.cleanupChooseLines

PurchFormletterParmDataInvoice.copyMarkupFromPurchOrder

PurchFormletterParmDataInvoice.processAdditional

PurchFormletterProvider.checkLines

PurchInvoiceJournalCreate.checkInvoice

PurchInvoiceJournalPost.postInventory

PurchLine.delete

PurchLine.setProjSalesPrice

PurchLine.update

PurchLine.validateWrite_Server

PurchLineCopy.canClose

PurchLineType.syncSalesLine

PurchLineType.updateInventory

PurchLineType.updatePurchStatus

PurchLineType.validateDelete

PurchPackingSlipJournalPost.postInventory

PurchPackingSlipJournalPost.updateSalesLine

PurchPackingSlipJournalPost.updateSalesTable
<table>
<thead>
<tr>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>PurchPackingSlipJournalPost.updateSourceLine</td>
</tr>
<tr>
<td>PurchQuantity_PackingSlip.calcQtyInvent</td>
</tr>
<tr>
<td>PurchQuantity_PackingSlip.calcQtyPurch</td>
</tr>
<tr>
<td>PurchReqAddItems.init</td>
</tr>
<tr>
<td>PurchReqTable.init</td>
</tr>
<tr>
<td>PurchReqWFExpendiParticipantProvider.resolve</td>
</tr>
<tr>
<td>PurchSelectLinesManager.mark</td>
</tr>
<tr>
<td>PurchTableForm.purchLine_WritePreSuper</td>
</tr>
<tr>
<td>PurchTableInteractionHelper.getDeliveryScheduleEnabled</td>
</tr>
<tr>
<td>PurchTableInteractionHelper.getHasMultipleDeliveries</td>
</tr>
<tr>
<td>PurchUpdateRemain.updateRemainPhysical</td>
</tr>
<tr>
<td>ReqCalc.insertItemInventSum</td>
</tr>
<tr>
<td>ReqTransPlanIdFilter.setPlanIdOnQueryRange</td>
</tr>
<tr>
<td>ReqTransPoMarkFirm.createPurchLine</td>
</tr>
<tr>
<td>ReqTransPoMarkFirm.createPurchTable</td>
</tr>
<tr>
<td>ReqTransPoMarkFirm.createPurchTable</td>
</tr>
<tr>
<td>ReqTransPoMarkFirm.dialog</td>
</tr>
<tr>
<td>RetailAssortmentLookupTask.CreateChannelGroups</td>
</tr>
<tr>
<td>RetailKitAssemblyOrder.createOrUpdateBOMJournal</td>
</tr>
<tr>
<td>RetailPackage.close</td>
</tr>
<tr>
<td>RetailProductPropertyManager.saveProductDimensions</td>
</tr>
<tr>
<td>RetailStatementPostChecker::checkStatement</td>
</tr>
<tr>
<td>ReturnTable.isDispositionCodeValid</td>
</tr>
<tr>
<td>RouteCopyToRoute</td>
</tr>
<tr>
<td>SalesCalcAvailableDlvDates.newCommonSalesDlvDateType</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>SalesCalcAvailableDlvDates.newCommonSalesDlvDateTypeByEntity</td>
</tr>
<tr>
<td>SalesCopying.copyLines</td>
</tr>
<tr>
<td>SalesEditLines.FormDesign/FormMenuFunctionButtonDownControl/ButtonPaymentSched/Method/clicked</td>
</tr>
<tr>
<td>SalesFormLetter.onCreatePaymSched</td>
</tr>
<tr>
<td>SalesFormLetterParmDataPackingSlip.selectChooseLines</td>
</tr>
<tr>
<td>SalesFormletterProvider.checkLines</td>
</tr>
<tr>
<td>SalesFormletterProvider.checkSalesLineChanged</td>
</tr>
<tr>
<td>SalesInvoiceController.initFormLetterReport</td>
</tr>
<tr>
<td>SalesInvoiceJournalPost.postInventory</td>
</tr>
<tr>
<td>SalesInvoiceJournalPostProj.updateInventory</td>
</tr>
<tr>
<td>SalesLine.createAlternativeItem</td>
</tr>
<tr>
<td>SalesLine.delete</td>
</tr>
<tr>
<td>SalesLine.returnLineUpdate</td>
</tr>
<tr>
<td>SalesLineCopy.canClose</td>
</tr>
<tr>
<td>SalesLineType.initFromCustInvoiceTrans</td>
</tr>
<tr>
<td>SalesOrderEntryStatistics.createOrderEntry</td>
</tr>
<tr>
<td>SalesOrderEntryStatistics.deleteOrderEntry</td>
</tr>
<tr>
<td>SalesPackingSlipDPItemId</td>
</tr>
<tr>
<td>SalesPackingSlipJournalPost.postInventory</td>
</tr>
<tr>
<td>SalesParmLine.setQty</td>
</tr>
<tr>
<td>SalesQuantity_Invoice.calcQtyInvent</td>
</tr>
<tr>
<td>SalesQuantity_Invoice.calcQtySales</td>
</tr>
<tr>
<td>SalesQuotationEditLinesForm_Sales_Confir method createSalesLines</td>
</tr>
<tr>
<td>SalesQuotationEditLinesForm_Sales_Confir method createSalesTable</td>
</tr>
<tr>
<td>SalesQuotationEditLinesForm_Sales_Confir createSalesLine</td>
</tr>
</tbody>
</table>
SalesQuotationEditLinesForm_Sales_Confir.createSalesLines
SalesQuotationLine.createLine
SalesQuotationLine.createQuotationLineFromTemplate
SalesQuotationLine.createQuotationLineFromTemplate
SalesQuotationLine.createQuotationLineFromTemplate
SalesQuotationLine.delete
SalesQuotationLine.setCostSalesPrice
SalesQuotationLine.updateSalesQuotationTable
SalesQuotationListPageInteraction.initIsSalesQuotation
SalesQuotationListPageInteraction.setButtonFollowup
SalesQuotationListPageInteraction.setButtonQuotation
SalesQuotationTableForm_DlvSchedule.updateSalesQuotationLineTable
SalesQuotationTableType.validateDelete
SalesTable.update
SalesTableForm_DeliverySchedule.updateSalesLineTable
SalesTableForm_ProjectSalesItem.resetSalesLine
SalesTableInteractionHelper.isOpenOrderNotReturnNotProjectRelatedSalesLine
SalesTableInteractionHelper.notPartiallyPickedPackedOrInvoiced
SalesTableInteractionHelper.notReservedOrderedNorPhysical
SalesTableType.checkUpdate
SalesTableType.checkUpdate
SalesTableType.checkUpdate
SalesTaxDeclarationInformationReportService.processReport
SettlementPair_Vend.fetchPayment
smmActivityParentLinkTable.insert
<table>
<thead>
<tr>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>smmBusRelTable.convert2Customer</td>
</tr>
<tr>
<td>smmBusRelTable.convert2Customer</td>
</tr>
<tr>
<td>SmmBusRelTable.convert2Vendor</td>
</tr>
<tr>
<td>SmmBusRelTable.convert2Vendor</td>
</tr>
<tr>
<td>smmBusRelTable.createConvertedBusRel</td>
</tr>
<tr>
<td>smmBusRelTable.createConvertedBusRel</td>
</tr>
<tr>
<td>smmLeadTable.createBusRelRecord</td>
</tr>
<tr>
<td>SmmProjectCreate.createProjectGroup</td>
</tr>
<tr>
<td>SpecTransInsertSetManager.insertDatabase</td>
</tr>
<tr>
<td>SubledgerJournalizerProjectExtension.createProjectActualCostDetail, createProjectActualHeader</td>
</tr>
<tr>
<td>SubledgerJournalizerProjectExtension.getProjectActualMap</td>
</tr>
<tr>
<td>SuppItemSales.initSalesLine</td>
</tr>
<tr>
<td>Table PurchLine.insert</td>
</tr>
<tr>
<td>Table PurchLine.insert</td>
</tr>
<tr>
<td>Table PurchLine.update</td>
</tr>
<tr>
<td>Table PurchTable.update</td>
</tr>
<tr>
<td>Table PurchTable.update</td>
</tr>
<tr>
<td>Table PurchTable.update</td>
</tr>
<tr>
<td>Table PurchTable.update</td>
</tr>
<tr>
<td>Table SalesLine.insert</td>
</tr>
<tr>
<td>table SalesLine.projIdModified</td>
</tr>
<tr>
<td>table SalesLine.salesQtyModifiedInteraction</td>
</tr>
<tr>
<td>Table SalesLine.update</td>
</tr>
<tr>
<td>Table SalesQuotationLine.insert</td>
</tr>
<tr>
<td>Table SalesQuotationLine.insert</td>
</tr>
<tr>
<td>Table SalesQuotationLine.update</td>
</tr>
<tr>
<td>Table SalesQuotationTable.update</td>
</tr>
<tr>
<td>Table SalesQuotationTable.update</td>
</tr>
<tr>
<td>Table SalesQuotationTable.update</td>
</tr>
<tr>
<td>Table SalesTable.update</td>
</tr>
</tbody>
</table>
 METHOD

Table SalesTable.update
Table\VendTable.name
Table\VendTable.updateOnHold
Tax.createOrphanLinkInsteadPost_RU
Tax.distributeTotalTax
Tax.distributeTotalTax
Tax.InsertIntersection
Tax.insertIntersection
Tax.post
Tax.Post
Tax.postCharge
Tax.postTaxProjInvoice_IN
Tax.postTaxPurchInvoice_IN
Tax.postTaxSalesInvoice_IN
Tax.saveAndPost
Tax.taxTotals
Tax.taxTotals
Tax.taxTotals
Tax.taxTotalsPosted
Tax.taxTotalsPosted
Tax.taxTotalsPosted
Tax.taxTotalsPosted
TaxBookSection.checkTaxBookSection
TaxCalculationAdjustment.calcManualInserted
TaxCalculationJournal.saveTaxTransfers
TaxFreeInvoice_Invoice.updateAndPost
<table>
<thead>
<tr>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>TaxInvoiceSpec parmTaxSpec</td>
</tr>
<tr>
<td>TaxListDP.insertTaxListTaxTmpData</td>
</tr>
<tr>
<td>TaxListDP.updateTaxListTaxTmpData</td>
</tr>
<tr>
<td>TaxMainAccDimensionListProvider populateMainAccountDimensionList</td>
</tr>
<tr>
<td>TaxPost.postToLedger</td>
</tr>
<tr>
<td>TaxReportController_US.init</td>
</tr>
<tr>
<td>TaxReportInclAdjustmentDP.insertTaxReportInclAdjustmentTmp</td>
</tr>
<tr>
<td>TaxReportingDP.insertTaxReportingTmp</td>
</tr>
<tr>
<td>TaxReverse.adjustTaxTransDueToExchangeRateGainLoss</td>
</tr>
<tr>
<td>TaxReverse.postAccountingCurrency</td>
</tr>
<tr>
<td>TaxReverse.postAccountingCurrency</td>
</tr>
<tr>
<td>TaxReverse.postAccountingCurrency</td>
</tr>
<tr>
<td>TaxReverse.postCharge</td>
</tr>
<tr>
<td>TaxReverse.postGainLossInReportingCurrency</td>
</tr>
<tr>
<td>TaxSales.calc</td>
</tr>
<tr>
<td>TaxSales.calc</td>
</tr>
<tr>
<td>TaxSales.calcMarkup</td>
</tr>
<tr>
<td>TaxSales.configureTaxForSalesLine</td>
</tr>
<tr>
<td>TaxVoucherService.taxAmountForLedgerType</td>
</tr>
<tr>
<td>TmpCustVendTrans.CustTransBalanceCurrency</td>
</tr>
<tr>
<td>TmpProjAdjustment.adjustmentType2JournalType</td>
</tr>
<tr>
<td>TmpProjAdjustment.adjustmentType2TransType</td>
</tr>
<tr>
<td>TmpProjAdjustment.updateFundingLimits</td>
</tr>
<tr>
<td>TmpProjAdjustmentCreate.salesPrice</td>
</tr>
<tr>
<td>TrvExpenseLinesVisibilityController.isVisibilityResetRequired</td>
</tr>
</tbody>
</table>
TrvExpenses.updateFormVisibilityOnCategoryChange
TrvExpTrans.calcTaxAmount
TrvTaxExpense.calculateTax
TSTimesheetEntry.init, initFields, setHeaderObjects, validateWrit, lookup
UnitOfMeasureConverter.Convert
VendInvoicePaymentAuthorizationTask.postSavedInvoice
VendPackingSlipTrans.unpostedInvoicePurchQtyServer and VendPackingSlipTrans.unpostedInvoiceInventQtyServer
VendTable.checkVATNumUsed
VendTax1099Update.calcMiscChargeAmountTax
VendTrans.documentDateModified
VendTrans.documentDateModified
VendVoucher.createTransOpen
VendVoucher.createTransOpen
WHSContainerTable.closeContainer
WHSDocumentRouting.translate
WHSLoadLine.orderHeader
WHSLoadTable.validateInventTransTypeMatches
WHSLoadTableAssignOriginInfo.classDeclaration
WmsArrivalCreateJournal.createWMSJournalTransFromTmp
WMSOrderTrans.split
WmsOrderTransType_Output.updateReservations
WorkflowHierarchyProviderHelperEventHandler::getPersonnelNumberIdBySysDictTypeDelegate
WorkTimeTable.removeDisplayCache
WrkCrtResourceAbilityMapController.insertData
Enable increase of decimal precision through extensibility for quantities
Enumerations made extensible

These enumerations have been made extensible in this update.

<table>
<thead>
<tr>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssetPostValue</td>
</tr>
<tr>
<td>AssetPropertyType</td>
</tr>
<tr>
<td>AssetStatus</td>
</tr>
<tr>
<td>CaseStatus</td>
</tr>
<tr>
<td>CommissionSalesRepCode</td>
</tr>
<tr>
<td>DirSubNameSequenceType</td>
</tr>
<tr>
<td>FreightSlipType</td>
</tr>
<tr>
<td>HRPLimitDocumentType</td>
</tr>
<tr>
<td>IntrastatPaymentMethod_IT</td>
</tr>
<tr>
<td>InventBlockingType</td>
</tr>
<tr>
<td>JmgPaySpecTypeEnumPick</td>
</tr>
<tr>
<td>KMQuestionAnswerInputType</td>
</tr>
<tr>
<td>LedgerJournalType</td>
</tr>
<tr>
<td>LogisticsAddrZipCodeImportCountryRegion</td>
</tr>
<tr>
<td>LogType</td>
</tr>
<tr>
<td>MarkupModuleType</td>
</tr>
<tr>
<td>MarkupModuleType</td>
</tr>
<tr>
<td>MCRBrokerValueType</td>
</tr>
<tr>
<td>PaymentType</td>
</tr>
<tr>
<td>PDSCalcElementTypeBase</td>
</tr>
<tr>
<td>PdsStatus</td>
</tr>
<tr>
<td>PdsVendorCheckItem</td>
</tr>
<tr>
<td>ProdStatus</td>
</tr>
<tr>
<td>ProjActiveAll</td>
</tr>
</tbody>
</table>
These SQL operations have been made extensible in this update.
ProjBudgetManager.createBudgetCostForecast
ProjBudgetManager.createBudgetEmplForecast
ProjBudgetManager.createBudgetRevenueForecast
ProjBudgetManager.createBudgetSalesForecast

**Metadata changes**

These metadata changes have been made in this update.

/Data entities/ReqPlannedOrderEntity.Is Public
/DataTypes/Extended Data Types/AmountQty.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/AssetDepreciationAmountUnitReportingCurrency.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/BOMProductQuantity.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/CostPriceNonMonetary.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/CostQuantity.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/MCRRoyaltyValue.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/PdsRebateValue.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/PriceDiscAmount.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/PriceQty.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/PriceUnit.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/PriceUnit.Scale
/DataTypes/Extended Data Types/ProductQuantity.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/ProductQuantityHourValue.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/ProjName.Extends
/DataTypes/Extended Data Types/TAMRebateValue.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/UnitAmountCur.NoOfDecimalsIsExtensible
/DataTypes/Extended Data Types/UnitAmountMST.NoOfDecimalsIsExtensible
Additional extensibility enhancements

In addition to the refactored methods, the following extensibility enhancements have been made.

- Bug request: "CustCollectionLetterTrans -> CollectionLetterNum" Relation properties
- Enable increase of decimal precision through extensibility for prices
- Enable increase of decimal precision through extensibility for weights
- Map Extension: LogisticsEntityLocationMap
- OMOperatingUnit should provide user friendly and defined value for DimAttributeOMDepartment.Value
- Redesign how InventPosting finds LedgerDimension
- Refactor WhsWorkExecuteDisplayAdjustIn to ProcessGuide framework
- Refactor WHSWorkExecuteDisplayChangeWarehouse to ProcessGuide framework
- Refactor WhsWorkExecuteDisplayInquiryItem to ProcessGuide framework
- Refactor WhsWorkExecuteDisplayInquiryLocation to ProcessGuide framework
- Refactor WhsWorkExecuteDisplayInquiryLP to ProcessGuide framework
- Refactor whsWorkExecuteDisplayReprintLabel to ProcessGuide framework
- Retail channel: Support BankDropOperationRequest
This is a list of extensibility features that were implemented in Dynamics 365 for Finance and Operations update 8.0.4. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Refactored methods to support extensibility

These methods have been refactored to support extensibility through chain of command, delegates, or by providing access to members.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgreementHeader.getModuleType</td>
<td>AssetSplit.construct</td>
</tr>
<tr>
<td>BankDepositSlipController.main</td>
<td>BankPositivePayExport.sendFileToUser</td>
</tr>
<tr>
<td>CaseDetailForm.getRecordsFromDataSource</td>
<td>CostSheetDesigner.DataSource:CostSheetCalculationFactor.validateWrite</td>
</tr>
<tr>
<td>CostSheetNodeCalculation.validate</td>
<td>CostSheetNodeCalculationRate.calcLowestLevel</td>
</tr>
<tr>
<td>CostSheetNodeCalculationSurcharge.equal</td>
<td>CustAccountStatementExtController.processParty</td>
</tr>
<tr>
<td>CustAccountStatementExtController.processParty</td>
<td>CustAccountStatementExtDInsertNewRecords</td>
</tr>
<tr>
<td>CustAccountStatementExtDSetSysDocuBrandDetails</td>
<td>CustBillOfExchangePost.postNextStep</td>
</tr>
<tr>
<td>CustBillOfExchangePostProtestHonored.postNextStep</td>
<td>CustCollectionJourController.runPrintMgmt</td>
</tr>
<tr>
<td>CustCollectionLetterCreate.createJournal</td>
<td>CustCollectionLetterCreate.run</td>
</tr>
<tr>
<td>CustCollectionLetterNote.CustCollectionLetterJour.active</td>
<td></td>
</tr>
<tr>
<td>METHOD</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>CustCollectionLetterPost.processRow</td>
<td></td>
</tr>
<tr>
<td>CustCollectionLetterPost.validateCollectionLetter</td>
<td></td>
</tr>
<tr>
<td>CustFreeInvoiceCorrection.createInvoiceLines</td>
<td></td>
</tr>
<tr>
<td>CustInterestPost.main</td>
<td></td>
</tr>
<tr>
<td>CustInterestPost.validateInterestTrans</td>
<td></td>
</tr>
<tr>
<td>CustInvoiceJour.printJournal</td>
<td></td>
</tr>
<tr>
<td>CustInvoiceTable.calcDue</td>
<td></td>
</tr>
<tr>
<td>CustOpenBalanceCurrency.Data Sources – VendTrans – init</td>
<td></td>
</tr>
<tr>
<td>CustOpenTrans.editMarkTrans</td>
<td></td>
</tr>
<tr>
<td>CustPackingSlipJourFormHelper.isCancelCorrectButtonsEnabled</td>
<td></td>
</tr>
<tr>
<td>CustPaymEntry.editIsMarkedForSettlement</td>
<td></td>
</tr>
<tr>
<td>CustPostInvoice.main</td>
<td></td>
</tr>
<tr>
<td>CustPostInvoice.run</td>
<td></td>
</tr>
<tr>
<td>CustPostInvoice.validate</td>
<td></td>
</tr>
<tr>
<td>CustPostInvoiceJob.custPostInvoiceUpdate</td>
<td></td>
</tr>
<tr>
<td>CustPostInvoiceJob.initializeCustPostProcess</td>
<td></td>
</tr>
<tr>
<td>CustPostInvoiceJob.main</td>
<td></td>
</tr>
<tr>
<td>CustPostInvoiceJob.processCustPostInvoiceUpdate</td>
<td></td>
</tr>
<tr>
<td>CustProvisionalBalanceDP.calculateAmounts</td>
<td></td>
</tr>
<tr>
<td>CustProvisionalBalanceDP.insertCustProvisionalBalanceTmp</td>
<td></td>
</tr>
<tr>
<td>CustProvisionalBalanceDP.populateCustProvisionalBalanceTmpProcessing</td>
<td></td>
</tr>
<tr>
<td>CustProvisionalBalanceDP.processReport</td>
<td></td>
</tr>
<tr>
<td>CustProvisionalBalanceDP.translateMainAccountNamesOnCustProvisionalBalanceTmpProcessing</td>
<td></td>
</tr>
<tr>
<td>CustQuotationJournal.launchReport</td>
<td></td>
</tr>
<tr>
<td>CustSettlementPriorityProcessing .createTempData</td>
<td></td>
</tr>
<tr>
<td>CustSettlementPriorityProcessing .setPaymentAmount</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>CustSettlementPriorityProcessing.updatePartialTrans</td>
<td></td>
</tr>
<tr>
<td>CustSettlementPriorityProcessing.classDeclaration</td>
<td></td>
</tr>
<tr>
<td>CustSettlementPriorityProcessing.constructCustOpenTrans</td>
<td></td>
</tr>
<tr>
<td>CustSettlementPriorityProcessing.constructCustPaymEntry</td>
<td></td>
</tr>
<tr>
<td>CustSettlementPriorityProcessing.constructOffsetVoucherCust</td>
<td></td>
</tr>
<tr>
<td>CustSettlementPriorityProcessing.getBillingPriorities</td>
<td></td>
</tr>
<tr>
<td>CustSettlementPriorityProcessing.getSettlementQuery</td>
<td></td>
</tr>
<tr>
<td>CustSettlementPriorityProcessing.initCustTransOpen</td>
<td></td>
</tr>
<tr>
<td>CustSettlementPriorityProcessing.insertAllLinesAccrossInvoices</td>
<td></td>
</tr>
<tr>
<td>CustSettlementPriorityProcessing.markTransactions</td>
<td></td>
</tr>
<tr>
<td>CustSettlementPriorityProcessing.markTransByCreditNoteOnBillingClasses</td>
<td></td>
</tr>
<tr>
<td>CustSettlementPriorityProcessing.validateMarkedTransactionOpenTrans</td>
<td></td>
</tr>
<tr>
<td>CustStatisticsUS - method calcStatistics</td>
<td></td>
</tr>
<tr>
<td>CustTable.validateCNPICPF_BR</td>
<td></td>
</tr>
<tr>
<td>CustTrans.checkReversal</td>
<td></td>
</tr>
<tr>
<td>CustVendCheque.processChequeNum</td>
<td></td>
</tr>
<tr>
<td>CustVendCreatePaymJournal_Vend.searchTransactions</td>
<td></td>
</tr>
<tr>
<td>CustVendExchAdjPostingEngine.addExchangeAdjustment</td>
<td></td>
</tr>
<tr>
<td>CustVendFindSettlements.getTmpTrans</td>
<td></td>
</tr>
<tr>
<td>CustVendPaymProposalTransferToJournal.transferProposalLineToJournal</td>
<td></td>
</tr>
<tr>
<td>CustVendSettle.createSummaryAccountReliefTransactions</td>
<td></td>
</tr>
<tr>
<td>CustVendSettle.mustOffsetOriginalSummaryDistributions</td>
<td></td>
</tr>
<tr>
<td>CustVendSettle.postingProfileSettle_CreateDistributions</td>
<td></td>
</tr>
<tr>
<td>CustVendSettle.settleNow</td>
<td></td>
</tr>
<tr>
<td>CustVendTransDistributionController.getDistributionFactorsForPostingTypes</td>
<td></td>
</tr>
<tr>
<td>CustVendTransExchAdjDistController.getDistributionFactorsForPostingTypes</td>
<td></td>
</tr>
<tr>
<td>METHOD</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td>CustVendTransSettle.post</td>
<td></td>
</tr>
<tr>
<td>CustVendVoucher.initLedgerPosting</td>
<td></td>
</tr>
<tr>
<td>CustWriteOff.createInterestWriteOffJournalForInterestTrans</td>
<td></td>
</tr>
<tr>
<td>DataFileImportExportUtils.readStreamWriterAndWriteToStreamWriter</td>
<td></td>
</tr>
<tr>
<td>EcoResProductCreate.initDefaultControlValues</td>
<td></td>
</tr>
<tr>
<td>EcoResProductReleaseManager.releaseToLegalEntity</td>
<td></td>
</tr>
<tr>
<td>ExchangeRateImportOperation.saveRates</td>
<td></td>
</tr>
<tr>
<td>FormletterJournalCreate.newPurchJournalCreate</td>
<td></td>
</tr>
<tr>
<td>FulfillmentLineView.NewExtension</td>
<td></td>
</tr>
<tr>
<td>InterCompanyPost.formLetterCollect</td>
<td></td>
</tr>
<tr>
<td>InventCostIndirectFinancial.remainingQty</td>
<td></td>
</tr>
<tr>
<td>InventCostItemDim.load</td>
<td></td>
</tr>
<tr>
<td>InventCostJournalIndirectCost &gt; addTrans</td>
<td></td>
</tr>
<tr>
<td>InventCostTrans.setRefTypeFromInventTransType</td>
<td></td>
</tr>
<tr>
<td>InventDimCtrl_Rep_Sales.initDimParmFormletter</td>
<td></td>
</tr>
<tr>
<td>InventDimCtrl_Rep_Sales.mustShowField</td>
<td></td>
</tr>
<tr>
<td>InventDimCtrl_Rep_Sales.reportStrItemId</td>
<td></td>
</tr>
<tr>
<td>InventDistinctProductOrderDefaultingController.construct</td>
<td></td>
</tr>
<tr>
<td>InventItemLocationCountingStatus.updateStopCountingJournal</td>
<td></td>
</tr>
<tr>
<td>InventItemPriceActivationJob.activateCostSheetCalculationFactor</td>
<td></td>
</tr>
<tr>
<td>InventJournalCheckPost_Movement.postTransLedgerMovement</td>
<td></td>
</tr>
<tr>
<td>InventJournalFormTrans_Movement.initReleasedProductSpecificDefaulting</td>
<td></td>
</tr>
<tr>
<td>InventoryMainAccDimensionListProvider.ledgerPostingType2InventAccountType</td>
<td></td>
</tr>
<tr>
<td>InventQualityOrderTable.setTestResult</td>
<td></td>
</tr>
<tr>
<td>InventSerial.init</td>
<td></td>
</tr>
<tr>
<td>InventSumPhysicalSpec.setValueQty</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>InventTable.insertInventItemOrderSetup</td>
<td></td>
</tr>
<tr>
<td>InventTrans.updateSumUp</td>
<td></td>
</tr>
<tr>
<td>InventTransferLine.updates</td>
<td></td>
</tr>
<tr>
<td>InventTransferUpd.beginLedger</td>
<td></td>
</tr>
<tr>
<td>InventTransIdSum.update</td>
<td></td>
</tr>
<tr>
<td>InventUpd_Estimated.updateFieldsChange</td>
<td></td>
</tr>
<tr>
<td>InventUpd_Financial.updateFinancialIssue</td>
<td></td>
</tr>
<tr>
<td>InventUpd_Financial.updateNow</td>
<td></td>
</tr>
<tr>
<td>InventUpd_Physical.updatePhysicalReturnedReceipt</td>
<td></td>
</tr>
<tr>
<td>InventUpd_Registered.pickReleatedIssueTransMore</td>
<td></td>
</tr>
<tr>
<td>InventUpd_Reservation.updateReserveMore</td>
<td></td>
</tr>
<tr>
<td>InventUpdateMarking.updateReservations</td>
<td></td>
</tr>
<tr>
<td>JmgAbsenceCalendar</td>
<td></td>
</tr>
<tr>
<td>JmgMESSwitchCode.init</td>
<td></td>
</tr>
<tr>
<td>LedgerExchAdj.postAdjustment</td>
<td></td>
</tr>
<tr>
<td>LedgerExchAdj.run</td>
<td></td>
</tr>
<tr>
<td>LedgerJournalEngine.initTaxGroup</td>
<td></td>
</tr>
<tr>
<td>LedgerJournalEngine_CustBillSettle.initCustOffsetAccount</td>
<td></td>
</tr>
<tr>
<td>LedgerJournalTransCustPaym.LedgerJournalTrans.CustVendBankAccountId.jumpRef</td>
<td></td>
</tr>
<tr>
<td>LedgerJournalTransUpdateCust</td>
<td></td>
</tr>
<tr>
<td>LedgerJournalTransUpdateLedger.updateNow</td>
<td></td>
</tr>
<tr>
<td>LogisticsPostalAddress.whsAddressFormatValidation</td>
<td></td>
</tr>
<tr>
<td>LogisticsPostalAddressFormEventHandler.updatePrimaryControl</td>
<td></td>
</tr>
<tr>
<td>Markup.calc</td>
<td></td>
</tr>
<tr>
<td>MCRCustPaym.ValidateWrite</td>
<td></td>
</tr>
<tr>
<td>McrCustPaymTotals_Sales.allPaymentsSubmitted</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>OffsetVoucher.updateNow</td>
<td></td>
</tr>
<tr>
<td>PmfCoByProdCalcTrans.updateRealCalcIndirect</td>
<td></td>
</tr>
<tr>
<td>PmfCoByProdCalcTrans.updateRealCalcIndirect</td>
<td></td>
</tr>
<tr>
<td>POSAPIdts.New Trigger</td>
<td></td>
</tr>
<tr>
<td>PriceDiscAdmCheckPost.checkForOverlapsAndGaps</td>
<td></td>
</tr>
<tr>
<td>PriceDiscAdmTrans.canEditPriceDiscValueField</td>
<td></td>
</tr>
<tr>
<td>ProcCategoryHierarchyManagement.init</td>
<td></td>
</tr>
<tr>
<td>ProdCalcTrans &gt; method updateRealCalcIndirect</td>
<td></td>
</tr>
<tr>
<td>ProdIndirectTrans &gt; method type2ItemCalcType</td>
<td></td>
</tr>
<tr>
<td>ProdJournalCheckPostProd.postTransLedger</td>
<td></td>
</tr>
<tr>
<td>ProdTableForm.handleProdTableCreatePreSuper</td>
<td></td>
</tr>
<tr>
<td>ProdUpdReportFinished.updateBOMConsumption</td>
<td></td>
</tr>
<tr>
<td>ProjAdjustmentUpdate.deleteJournal</td>
<td></td>
</tr>
<tr>
<td>ProjFundingLimitTrackingManager.updateUsingSourceDocument</td>
<td></td>
</tr>
<tr>
<td>ProjInvoiceProposalInsertLines.run</td>
<td></td>
</tr>
<tr>
<td>ProjInvoiceTableCreate.canClose</td>
<td></td>
</tr>
<tr>
<td>ProjInvoiceTableCreate.initializeValues</td>
<td></td>
</tr>
<tr>
<td>ProjPlanVersionsManager.createTemplateHierarchy</td>
<td></td>
</tr>
<tr>
<td>ProjPostCostJournal.projTransCreate</td>
<td></td>
</tr>
<tr>
<td>ProjSalesItemReq.SalesLine.linkActive</td>
<td></td>
</tr>
<tr>
<td>ProjTableType_TimeMaterial.validateWrite</td>
<td></td>
</tr>
<tr>
<td>PurchAgreement.applyQueryRanges</td>
<td></td>
</tr>
<tr>
<td>PurchApproveJournalPost.postPurgeLedgerAccount</td>
<td></td>
</tr>
<tr>
<td>PurchaseOrderResponseService.shouldPurchaseOrderBeAutoConfirmed</td>
<td></td>
</tr>
<tr>
<td>PurchAutoCreate_PurchReq.initializeAndCreatePurchLine</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>PurchAutoCreate_PurchReq.initializeAndCreatePurchLine</td>
<td></td>
</tr>
<tr>
<td>PurchAutoCreate_Sales.createLine</td>
<td></td>
</tr>
<tr>
<td>PurchCalcTax.construct</td>
<td></td>
</tr>
<tr>
<td>PurchCancel.run</td>
<td></td>
</tr>
<tr>
<td>PurchCreateFromOrder.insertMinMaxQty</td>
<td></td>
</tr>
<tr>
<td>PurchCreateFromSalesOrder.insertIntoTmpPurchLinePrice</td>
<td></td>
</tr>
<tr>
<td>PurchCreateFromSalesOrder.refreshCallerDataSource</td>
<td></td>
</tr>
<tr>
<td>PurchCreateFromSalesOrderSalesline.specifyMinMaxQty</td>
<td></td>
</tr>
<tr>
<td>PurchCreateFromSalesOrderSalesLine.specifyVendAccount</td>
<td></td>
</tr>
<tr>
<td>PurchCreateFromSalesOrdervalidateSalesLine</td>
<td></td>
</tr>
<tr>
<td>PurchEditLines.canClose</td>
<td></td>
</tr>
<tr>
<td>PurchEditLinesForm.construct</td>
<td></td>
</tr>
<tr>
<td>PurchFormLetter.construct</td>
<td></td>
</tr>
<tr>
<td>PurchFormLetterContract.newFromPackedVersion</td>
<td></td>
</tr>
<tr>
<td>PurchFormletterParmDataInvoice.selectFromJournalLines</td>
<td></td>
</tr>
<tr>
<td>PurchFormLetterProvider.checkPurchLineChanged</td>
<td></td>
</tr>
<tr>
<td>PurchInvoiceJournalPost.updateSourceLine</td>
<td></td>
</tr>
<tr>
<td>PurchLine.checkInvoiceConstraints</td>
<td></td>
</tr>
<tr>
<td>PurchLine.ledgerDimensionItem</td>
<td></td>
</tr>
<tr>
<td>PurchLine.ledgerDimensionReceipt</td>
<td></td>
</tr>
<tr>
<td>Purchline.unLinkAgreementLinePrompt</td>
<td></td>
</tr>
<tr>
<td>PurchLine.validateField</td>
<td></td>
</tr>
<tr>
<td>PurchLine::setProjSalesPrice</td>
<td></td>
</tr>
<tr>
<td>PurchPrepayTable.updateAdvanceApplicationRemaining</td>
<td></td>
</tr>
<tr>
<td>PurchPurchOrderJournalPost.updateSourceTable</td>
<td></td>
</tr>
<tr>
<td>PurchReqCreate.init</td>
<td></td>
</tr>
</tbody>
</table>
PurchReqLine.setDefaultDimension
PurchReqLine.validateWrite
PurchReqTable.init
PurchReqTableForm.new
PurchRFQCaseTable.init
PurchTable.jumpRefIntercompanySalesOrder
PurchTableForm_DeliverySchedule.updatePurchLineTable
PurchTableInteraction.enableHeaderReceive
PurchTableType.validateDelete
PurchTableUpdateFromPurchReqLineMap.update
ReqTransPoMarkFirm.setPurchBuyerGroupid & updatePurchBuyerGroup
RetailGroupMemberLineHelper.internalCreateOrUpdateOrRemoveRetailGroupMemberLine
RetailLabelDP.insertTmpTable
SalesCalcAvailableDlvDates.mainOnServer
SalesConfirmJournalPost.updateSourceTable
SalesCopying.editCopy
SalesCopying.editMarkAll
SalesCopying.initReturnOrderFromCustomer
SalesCreateQuotation.canClose
SalesDropShipmentCancel.removeMarking
SalesEditLines.canClose
SalesFormletterParmData.reArrangeLines
SalesFormletterParmData.reArrangeSplit
SalesFormletterParmData.reArrangeSplit
SalesFormLetterProvider.checkJournal
SalesInvoiceDP.insertGiroInformation
<table>
<thead>
<tr>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>SalesInvoiceJournalCreate.checkDocumentData_PL</td>
</tr>
<tr>
<td>SalesInvoiceJournalPostBase.postLine</td>
</tr>
<tr>
<td>SalesLine.resetInvent</td>
</tr>
<tr>
<td>SalesLineType.initDimensionsSpecificDefaulting</td>
</tr>
<tr>
<td>SalesLineType.initReleasedProductSpecificDefaulting</td>
</tr>
<tr>
<td>SalesPackingSlipDP.setSalesPackingSlipDetailsTmp</td>
</tr>
<tr>
<td>SalesPackingSlipJournalCreate.updateJournalLine</td>
</tr>
<tr>
<td>SalesPackingSlipJournalCreate.updateJournalLine</td>
</tr>
<tr>
<td>SalesPackingSlipJournalPost.insertBackorderLine</td>
</tr>
<tr>
<td>SalesPackingSlipJournalPost.interCompanyPost</td>
</tr>
<tr>
<td>SalesPackingSlipJournalPost.updateJournalLine</td>
</tr>
<tr>
<td>SalesPurchSummaryModel_Account.createNewJournal</td>
</tr>
<tr>
<td>SalesQuotationCalcTax_Sales.construct</td>
</tr>
<tr>
<td>SalesQuotationEditLinesForm.postUpdate</td>
</tr>
<tr>
<td>SalesQuotationLineType.initFromProjTable</td>
</tr>
<tr>
<td>SalesQuotationLineType.initReleasedProductSpecificDefaulting</td>
</tr>
<tr>
<td>SalesQuotationTable.modifiedField</td>
</tr>
<tr>
<td>SalesQuotationTableForm.createFromTemplate</td>
</tr>
<tr>
<td>SalesQuotationUpdate_Cancelled.run</td>
</tr>
<tr>
<td>SalesQuotationUpdate_Lost.run</td>
</tr>
<tr>
<td>SalesTable.initFromCustTableMandatoryFields</td>
</tr>
<tr>
<td>SalesTable.jumpRefIntercompanyPurchaseOrder</td>
</tr>
<tr>
<td>SalesTable.setShipCarrierFromLogisticsLocation</td>
</tr>
<tr>
<td>SalesTable.update</td>
</tr>
<tr>
<td>SalesTableForm.interCompanyAutoCreateOrders</td>
</tr>
<tr>
<td>SettlementPair.createSettlementForDebitOrCreditTrans</td>
</tr>
<tr>
<td>METHOD</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>SmaServiceFunctionLine_Transfer.createJournalLine</td>
</tr>
<tr>
<td>SmaServiceFunctionLine_Transfer.postJournalTransType</td>
</tr>
<tr>
<td>SmaServiceOrderCreate.createServiceOrderLine</td>
</tr>
<tr>
<td>SmaSubscriptionGenerator::postTrans</td>
</tr>
<tr>
<td>smmBusRelTable.relation2Vendor</td>
</tr>
<tr>
<td>smmBusRelTable.updateQuotations</td>
</tr>
<tr>
<td>SmmCampaignBroadcast::validate</td>
</tr>
<tr>
<td>SmmOpportunityStatusUpdate.updateOpportunity</td>
</tr>
<tr>
<td>smmOpportunityTable(Methods)\openQuotation</td>
</tr>
<tr>
<td>SmmProjectCreate.createSingleProject</td>
</tr>
<tr>
<td>SmmProjectCreate.createSingleProject</td>
</tr>
<tr>
<td>SmmUpdateBusRel.updateFromVendTableSFA2</td>
</tr>
<tr>
<td>SubledgerJournalTransferController.run</td>
</tr>
<tr>
<td>SuppItem.calcSuppItem</td>
</tr>
<tr>
<td>SuppItem::newSuppItem</td>
</tr>
<tr>
<td>SuppItemCreate::newSuppItemCreate</td>
</tr>
<tr>
<td>Tax::post</td>
</tr>
<tr>
<td>Tax.saveAndPost</td>
</tr>
<tr>
<td>TaxFreeInvoice_Invoice.updateAndPost</td>
</tr>
<tr>
<td>TaxPost.moveTaxLineToNewOwner</td>
</tr>
<tr>
<td>TaxPost.saveAndPostFromTmpTaxWorkTrans</td>
</tr>
<tr>
<td>TaxPost.saveAndPostFromTmpTaxWorkTrans</td>
</tr>
<tr>
<td>TaxVoucherService.postTaxOnErrorAccount</td>
</tr>
<tr>
<td>TradePackingSlipJourChain.createRelationship</td>
</tr>
<tr>
<td>TradeTotals.calc</td>
</tr>
<tr>
<td>TradeTotals.updateOrderBalances</td>
</tr>
<tr>
<td>METHOD</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>VendAccountStatementIntDP.processReport</td>
</tr>
<tr>
<td>VendEditInvoice\DataSource\VendInvoiceInfoTable\Methods\write</td>
</tr>
<tr>
<td>VendInvoiceMatching.initExpectedValues</td>
</tr>
<tr>
<td>VendInvoiceWFParticipantProviderExpend.resolve</td>
</tr>
<tr>
<td>VendOpenTrans.editMarkTrans</td>
</tr>
<tr>
<td>VendorInvoiceLineSourceDocLineItem.calculateSourceDocumentAmountMap</td>
</tr>
<tr>
<td>VendPaymentJournalDP.insertDataFromLedgerJournalTrans</td>
</tr>
<tr>
<td>VendPaymentJournalDP.insertDataFromSpecTrans</td>
</tr>
<tr>
<td>VendPromissoryNotePost.postNextStep</td>
</tr>
<tr>
<td>VendProvisionalBalanceDP.calculateAmounts</td>
</tr>
<tr>
<td>VendProvisionalBalanceDP.insertVendProvisionalBalanceTmp</td>
</tr>
<tr>
<td>VendProvisionalBalanceDP.processReport</td>
</tr>
<tr>
<td>VendTransListDP.ProcessReport</td>
</tr>
<tr>
<td>VendVoucher.createInvoiceJournal</td>
</tr>
<tr>
<td>WHSDocumentRouting.printDocument</td>
</tr>
<tr>
<td>WhsLoadLineInventTransValidator.checkLoadLineInventTransConsistencyOnInventoryUpdate</td>
</tr>
<tr>
<td>WHSLoadLineUpdater.initAndInsertLoadLine</td>
</tr>
<tr>
<td>WhsShipConfirm.createASNItems</td>
</tr>
<tr>
<td>WhsWarehouseRelease.createLoadLines</td>
</tr>
<tr>
<td>WHSWorkExecute.executeShortPick</td>
</tr>
<tr>
<td>WHSWorkExecuteDisplayLPReceiving.displayForm</td>
</tr>
<tr>
<td>WHSWorkExecuteDisplayLPReceiving.displayNextForm</td>
</tr>
<tr>
<td>WHSWorkExecuteDisplayMovementByTemplate.displayForm</td>
</tr>
<tr>
<td>WsWorkExecuteForm.createLabel</td>
</tr>
<tr>
<td>WmsJournalCheckPostReception.postTrans</td>
</tr>
<tr>
<td>WmsOnlineCountingServer.getMovement</td>
</tr>
</tbody>
</table>
### Enumerations made extensible

These enumerations have been made extensible in this update.

<table>
<thead>
<tr>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABCModel</td>
</tr>
<tr>
<td>AccountOrder</td>
</tr>
<tr>
<td>AmountUnit</td>
</tr>
<tr>
<td>CostCalculationRateSubtype</td>
</tr>
<tr>
<td>CurrencyGainLossAccountType</td>
</tr>
<tr>
<td>CustInterestCodeSource</td>
</tr>
<tr>
<td>CustPaymentType</td>
</tr>
<tr>
<td>DirViewLocationNodeType</td>
</tr>
<tr>
<td>InterestCalcAccountChoice</td>
</tr>
<tr>
<td>InventTransPostingType</td>
</tr>
<tr>
<td>MCRCustSearchType</td>
</tr>
<tr>
<td>PaymentStub</td>
</tr>
<tr>
<td>PaymentStubInclAll</td>
</tr>
<tr>
<td>ProjCompletePrincip</td>
</tr>
<tr>
<td>ProjMatchingPrincip</td>
</tr>
<tr>
<td>RetailPOSSeedDataType</td>
</tr>
<tr>
<td>SalesQuotationStatus</td>
</tr>
<tr>
<td>TMSApptStatus</td>
</tr>
<tr>
<td>TMSCommunicationType</td>
</tr>
</tbody>
</table>

### Additional extensibility enhancements
In addition to the refactored methods, the following extensibility enhancements have been made.

- **CustVendSettle**: set variables protected instead of private.
- Data manipulation method not raising event: WHSLocationDirective.loopLocDirLines.
- Data manipulation method not raising event: WhsWorkCreate.createTempLine.
- Map extension: LogisticsPostalAddressMap.
- Map extension: PurchReqLineMap.
- Metadata change: DataEntityView/ProjWBSActivityEstimatesEntity.Is Public = Yes.
- Metadata change: DataEntityView/ProjWBSActivityEstimatesEntity.Public Collection Name = ProjWBSActivityEstimates.
- Metadata change: DataEntityView/ProjWBSActivityEstimatesEntity.Public Entity Name = ProjWBSActivityEstimates.
- Metadata change: Form/WHSLoadPlanningListPage/FormDesign/FormDesign/FormActionPaneTabControl/ActionPaneTabShipReceive.Needed permission.
- Metadata change: WHSContainerLine, Relations WHSLoadLine & WHSShipmentTable.On Delete.
- Metadata change: WHSContainerTable, Relation WHSShipmentTable.On Delete.
- Project pricing: complete uptake of new pricing find methods.
This is a list of extensibility features that were implemented in Dynamics 365 for Finance and Operations update 8.0.3. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Refactored methods to support extensibility

These methods have been refactored to support extensibility through chain of command, delegates, or by providing access to members.

<table>
<thead>
<tr>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccountingSourceExplorerProcessor.filterEntries</td>
</tr>
<tr>
<td>AgreementClassification.init</td>
</tr>
<tr>
<td>AgreementConfirm.createLineVolumeCommitmentHistory</td>
</tr>
<tr>
<td>AgreementConfirm.newAgreementConfirm</td>
</tr>
<tr>
<td>AgreementLine.findLineForAutoMatch</td>
</tr>
<tr>
<td>AgreementLine.getAgreementLinesForOrderLine</td>
</tr>
<tr>
<td>AgreementLine.getAgreementLinesForPurchReqLine</td>
</tr>
<tr>
<td>AgreementLine.getAgreementLinesList</td>
</tr>
<tr>
<td>Bank.FR.checkControlText</td>
</tr>
<tr>
<td>Bank.IT.checkCIN</td>
</tr>
<tr>
<td>Bank.IT.checkRegistrationNum</td>
</tr>
<tr>
<td>BankAccountTrans.insert</td>
</tr>
<tr>
<td>BankAccountTrans.update</td>
</tr>
<tr>
<td>BankChequeCopy.fillTmpChequePrintout</td>
</tr>
<tr>
<td>BankChequePrint.printDocument</td>
</tr>
<tr>
<td>BankPaymAdviceReportGeneratorVend</td>
</tr>
<tr>
<td>BankReconciliationMatchRuleLine.getFieldsOfSysGenMatchRuleLineOfDoc</td>
</tr>
<tr>
<td>METHOD</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BankReconciliationMatchRuleLine.getFieldsOfSysGenMatchRuleLineOfDoc</td>
</tr>
<tr>
<td>BankReconMatchingRuleAutoProcessor.getSearchedDocumentIdList</td>
</tr>
<tr>
<td>BankReconMatchingRuleAutoProcessor.getSearchedDocumentIdList</td>
</tr>
<tr>
<td>BankVoucher.createBankAccountTrans</td>
</tr>
<tr>
<td>BankVoucher.createBankAccountTrans</td>
</tr>
<tr>
<td>BomCopyToProd.copyTo</td>
</tr>
<tr>
<td>BudgetPlanLineFieldActiveViewMapping.getBudgetPlanLineFieldName</td>
</tr>
<tr>
<td>BudgetTransaction.openLinesInExcel</td>
</tr>
<tr>
<td>ChequeController.init</td>
</tr>
<tr>
<td>CustAccountStatementExt.main</td>
</tr>
<tr>
<td>CustAccountStatementExtController.includeOnStatement</td>
</tr>
<tr>
<td>CustAccountStatementExtController.insertCustAccountStatementExtTmp</td>
</tr>
<tr>
<td>CustAccountStatementExtController.setCommonData</td>
</tr>
<tr>
<td>CustAccountStatementExtController.tmpCustVendTrans</td>
</tr>
<tr>
<td>CustAccountStatementExtUIBuilder.build</td>
</tr>
<tr>
<td>CustAuditorDP.setCustAuditorTmp</td>
</tr>
<tr>
<td>CustCollectionJourDP.insertCustCollectionJourDP</td>
</tr>
<tr>
<td>CustCreditLimit.Balance</td>
</tr>
<tr>
<td>CustInterestNoteDp.processReport</td>
</tr>
<tr>
<td>CustInvoiceJourDP.printJournal</td>
</tr>
<tr>
<td>CustInvoiceTable.checkCreditLimit</td>
</tr>
<tr>
<td>CustPackingSlipJourDP.interCompanyUpdate</td>
</tr>
<tr>
<td>CustPaymEntry.updateConditionalControls</td>
</tr>
<tr>
<td>custPostInvoicejob.custPostInvoiceUpdate</td>
</tr>
<tr>
<td>CustTrans.reverseTransact</td>
</tr>
</tbody>
</table>
CustVendCheque.createBankChequePaymentTrans
CustVendCheque.createBankChequePaymentTrans
CustVendCheque.initTmpChequePrintout
CustVendCheque.output
CustVendCheque.output
CustVendChequeSlipTextCalculator.getChequeDocLength
CustVendSumForPaym.validateSEPATransaction
CustVendSumUpJournal.createTrans
CustVendVoucher.post
DimDerDistRuleProjectTimesheetsExt.populateDimAllocListIntercompany
DimDerJourRuleProjectTimesheetsExt.getDefaultDimensionAllocation
DirPartyVerification.selectionChanged
EcoResCategoryTreeDatasource.initializeAvailableCategoriesMap
EcoResProductCreate.writeMoreFields
EcoResProductDetailsExtended.initInventDimensionsMetadataEntries
ElectronicPaymentRemitExport_BR.construct
ForecastPuch
ForecastSales.accountConsumption
ForecastSales.accountDisc
ForecastSales.accountIssue
ForecastSales.accountSales
InventPosting.accountItemLedgerDimension
InventSupply.init
InventTrans.insertReturnTransOrigin
InventTransferParmLine - several methods
<table>
<thead>
<tr>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>InventTransferUpd::updateLines</td>
</tr>
<tr>
<td>InventTransFormHelper::formQueryAddDynalink</td>
</tr>
<tr>
<td>InventTransWMS_Pick::updateInventServer</td>
</tr>
<tr>
<td>InventUpd_Physical::updatePhysicalReceiptTrans</td>
</tr>
<tr>
<td>InventUpdate::writeInventTrans</td>
</tr>
<tr>
<td>InventUpdate::createInventTransOriginAndReferences</td>
</tr>
<tr>
<td>InventValueReportPopulateItem::findReportLine</td>
</tr>
<tr>
<td>JmgRegistration.JmgJobTable</td>
</tr>
<tr>
<td>JournalizingDefinitionManager.newJournalizingDefinitionManagerPurch</td>
</tr>
<tr>
<td>JournalStatic::initializeDataModel</td>
</tr>
<tr>
<td>LedgerFinancialJournalReportDPBE::calcDebCredTotals</td>
</tr>
<tr>
<td>LedgerFinancialJournalReportDPBE::processReport</td>
</tr>
<tr>
<td>LedgerJournalDP::insertJournalTransForLedgerJournalTable</td>
</tr>
<tr>
<td>LedgerJournalDP::insertLedgerJournalTmp</td>
</tr>
<tr>
<td>LedgerJournalEngine::newJournalActive</td>
</tr>
<tr>
<td>LedgerJournalTrans::checkAllowPosting</td>
</tr>
<tr>
<td>LedgerJournalTransUpdateBank::setBankVoucherSource</td>
</tr>
<tr>
<td>LedgerJournalTransUpdateBank::updateNow</td>
</tr>
<tr>
<td>LedgerJournalTransUpdateBankLC::addBankVoucher</td>
</tr>
<tr>
<td>LedgerPostingGeneralJournalController::transferLines</td>
</tr>
<tr>
<td>LedgerPurchaseJournalReportDPBE::insertIntoTempTable</td>
</tr>
<tr>
<td>LedgerSalesJournalReportDPBE::processReport</td>
</tr>
<tr>
<td>LedgerTransFurtherPosting::createLedgerJournalTransFromGenJour</td>
</tr>
<tr>
<td>LedgerTransVoucher::getSubledgerVoucherLinkDataSource</td>
</tr>
<tr>
<td>LedgerTransVoucher::getSubledgerVoucherLinkDataSource</td>
</tr>
<tr>
<td>METHOD</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>LedgerTransVoucher.getVoucherDateRange</td>
</tr>
<tr>
<td>LedgerVoucherObject.updateLedgerPostingJournal</td>
</tr>
<tr>
<td>LedgerVoucherTransObject.checkRounding</td>
</tr>
<tr>
<td>Markup.insertMarkupTrans</td>
</tr>
<tr>
<td>MarkupTrans.MarkupTable.MarkupCode.Lookup</td>
</tr>
<tr>
<td>PaymSchedCalc::init*</td>
</tr>
<tr>
<td>PaymSchedCalc_Line::createTransaction</td>
</tr>
<tr>
<td>PdsApprovedVendorListCheck.newBasedOnTableType</td>
</tr>
<tr>
<td>PmfFormulaCoBy.run</td>
</tr>
<tr>
<td>PmfFormulaCoBy.ValidateField</td>
</tr>
<tr>
<td>PmfProdCoBy.ValidateField</td>
</tr>
<tr>
<td>PmfProdCoBy.ValidateWrite</td>
</tr>
<tr>
<td>PriceDiscAdmSearch</td>
</tr>
<tr>
<td>PriceDiscPolicyDialog.runPolicyDialog</td>
</tr>
<tr>
<td>ProdBOM.checkIsItemsReleased</td>
</tr>
<tr>
<td>ProdBOM::update</td>
</tr>
<tr>
<td>ProdJournalProd.Insert</td>
</tr>
<tr>
<td>ProdPurch.createPurchTable</td>
</tr>
<tr>
<td>ProdUpdHistoricalCost_Process.checkValidCoBy</td>
</tr>
<tr>
<td>ProdUpdReportFinished::updateBOMConsumption</td>
</tr>
<tr>
<td>ProdUpdStartUp.getListOfBOMJournals</td>
</tr>
<tr>
<td>ProdUpdStatusDecrease_StartUp.reverseBOMStartUp</td>
</tr>
<tr>
<td>ProjBudgetParticipantProvider.resolveByProject</td>
</tr>
<tr>
<td>ProjBudgetParticipantProvider.resolveByProjectHierarchy</td>
</tr>
<tr>
<td>ProjBudgetParticipantProvider.resolveByRootProject</td>
</tr>
<tr>
<td>METHOD</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>ProjCaseActivitiesHandlers.mmmActivities.onValidatedDelete</td>
</tr>
<tr>
<td>ProjControlPeriod.forecast</td>
</tr>
<tr>
<td>ProjControlPeriod.forecast</td>
</tr>
<tr>
<td>ProjControlPeriodCostGroup.totalBudgetMinusActual</td>
</tr>
<tr>
<td>ProjControlPeriodCostGroup.totalBudgetMinusActual</td>
</tr>
<tr>
<td>ProjectPosting.costLedgerDimension.</td>
</tr>
<tr>
<td>ProjectPosting.getProjectLedgerDimension.</td>
</tr>
<tr>
<td>ProjForecastEmpInitValue</td>
</tr>
<tr>
<td>ProjForecastReduceHour.constructQuery</td>
</tr>
<tr>
<td>ProjFundingSource.setInvoiceLocation</td>
</tr>
<tr>
<td>ProjGroup.initFromProjType</td>
</tr>
<tr>
<td>ProjIntercompanyCustomerInvoiceCreator.createInvoiceLine</td>
</tr>
<tr>
<td>ProjIntercompanyTransactionSelection.runQuery</td>
</tr>
<tr>
<td>ProjIntercompanyTransQuery.buildExpenseQuery</td>
</tr>
<tr>
<td>ProjIntercompanyTransQuery.buildHoursQuery</td>
</tr>
<tr>
<td>ProjIntercompanyTransQuery.buildVendorInvoiceLinesQuery</td>
</tr>
<tr>
<td>ProjInventJournalTransMapForm.checkActivity</td>
</tr>
<tr>
<td>projInvoiceChoose.setProposalJour</td>
</tr>
<tr>
<td>ProjInvoiceChoose.doRevenue</td>
</tr>
<tr>
<td>ProjInvoiceChoose.updateInvoiceTotal</td>
</tr>
<tr>
<td>ProjInvoiceProposalCreateLines.isRevenueTrans</td>
</tr>
<tr>
<td>ProjInvoiceProposalCreateLinesBase.createProposalTrans</td>
</tr>
<tr>
<td>ProjInvoiceProposalCreateLinesBase.doOnAccount</td>
</tr>
<tr>
<td>ProjInvoiceTable</td>
</tr>
<tr>
<td>ProjLedger.classDeclaration</td>
</tr>
<tr>
<td>METHOD</td>
</tr>
<tr>
<td>---------------------------</td>
</tr>
<tr>
<td>ProjPostItemJournal.projTransCreate</td>
</tr>
<tr>
<td>ProjProjectsListPage.CtrlStages</td>
</tr>
<tr>
<td>ProjProjectsListPageInteraction.enableButton</td>
</tr>
<tr>
<td>ProjProjectsListPageInteraction.showButton</td>
</tr>
<tr>
<td>ProjStatusUpd.main</td>
</tr>
<tr>
<td>ProjStatusUpd.new</td>
</tr>
<tr>
<td>ProjTable - ProjTable datasource.write</td>
</tr>
<tr>
<td>ProjTable.clicked</td>
</tr>
<tr>
<td>ProjTable.editSubProj</td>
</tr>
<tr>
<td>ProjTable.editSubProj</td>
</tr>
<tr>
<td>ProjTableCreate.close</td>
</tr>
<tr>
<td>ProjTableCreate.run</td>
</tr>
<tr>
<td>ProjTableCreate.write</td>
</tr>
<tr>
<td>ProjTableCreate.write</td>
</tr>
<tr>
<td>ProjTableCreate.write</td>
</tr>
<tr>
<td>ProjTableLookup.ProjProjectLookup.init</td>
</tr>
<tr>
<td>PSAProjInvoiceDP.insertPSAProjInvoiceHeaderTmp</td>
</tr>
<tr>
<td>PSAProjInvoiceTaxTmp.insertPSAProjInvoiceTmpForTax</td>
</tr>
<tr>
<td>PsaProjProposalSelection</td>
</tr>
<tr>
<td>PurchAgreementAutoCreate::construct</td>
</tr>
<tr>
<td>PurchAutoCreate.setPurchTable</td>
</tr>
<tr>
<td>PurchAutoCreate_PurchReq.initializeAndCreatePurchLine</td>
</tr>
<tr>
<td>PurchAutoCreate_PurchReq.initializeAndCreatePurchLine</td>
</tr>
<tr>
<td>PurchAutoCreate_ReleaseFromAgreement.updateFinDimFromAgreemHeader</td>
</tr>
<tr>
<td>PurchCreateFromSalesOrder.shouldCreatePurchOrder</td>
</tr>
<tr>
<td>PurchFormLetter::main</td>
</tr>
<tr>
<td>METHOD</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>PurchFormLetter::main</td>
</tr>
<tr>
<td>PurchFormletterParmDataPackingSlip::reSelectLines</td>
</tr>
<tr>
<td>PurchFormletterParmDataPackingSlip::selectChooseLines</td>
</tr>
<tr>
<td>PurchFormletterParmDataPurchOrder::selectChooseLines</td>
</tr>
<tr>
<td>PurchInvoiceJournalPost.checkBeforePostingLine</td>
</tr>
<tr>
<td>PurchInvoiceJournalPost.updateSourceLine</td>
</tr>
<tr>
<td>Purchline.createline</td>
</tr>
<tr>
<td>PurchOrderLineBudgetControlPolicy.canCheckBudget</td>
</tr>
<tr>
<td>PurchReceiptsListDP.setPurchReceiptsListDetailsTmp</td>
</tr>
<tr>
<td>PurchReceiptsListDP.setPurchReceiptsListHeaderTmp</td>
</tr>
<tr>
<td>PurchRFQAceptJournalPost.updatePurchReq</td>
</tr>
<tr>
<td>ReqCalc.covCalcDim</td>
</tr>
<tr>
<td>ReqTrans.createTransferDemand</td>
</tr>
<tr>
<td>ReqTransPoMarkFirm.createProdRoute</td>
</tr>
<tr>
<td>RetailPeriodicDiscount.ClassDeclaration</td>
</tr>
<tr>
<td>RetailTransactionServiceOrders.cancelCustomerOrder</td>
</tr>
<tr>
<td>Return.ReturnDispositionCodeId::validate</td>
</tr>
<tr>
<td>SalesAutoCreate::construct</td>
</tr>
<tr>
<td>SalesFormLetter.mainOnServer</td>
</tr>
<tr>
<td>SalesFormLetter::main</td>
</tr>
<tr>
<td>SalesFormletterParmDataConfirm::selectChooseLines</td>
</tr>
<tr>
<td>SalesFormletterParmDataInvoice::mayJournalTransBePosted</td>
</tr>
<tr>
<td>SalesFormletterParmDataInvoice::selectChooseLines</td>
</tr>
<tr>
<td>SalesFormletterParmDataPackingslip::selectChooseLines</td>
</tr>
</tbody>
</table>
METHOD

TransactionReversal_Vend.reversal

TransactionReversal_Vend.reversal

TsTimesheetAddFavorites.addToFavorites

TsTimesheetCreate.createTimesheetLine

TSTimesheetEntry.initFields

TSTimesheetFavorites.createTimesheetLines

TSTimesheetLine.setCategoryIdFromActivity

VendInvoiceDocumentDP.insertVendInvoiceDocumentTmp

WHSLoadLine.validateStatus

WHSLoadLineAllocationProcessor.allocateLoadLine

WHSPostEngine.validateAnyDimAboveLocationMissing

WhsWarehouseRelease.createShipmentsForAllSalesOrders

WhsWarehouseRelease.createShipmentsForTransferOrders

WhsWorkCreateLP.createTempTable

WHSWorkCreateProdPut.createTempTable

WHSWorkExecuteDisplay.buildNextDimensionCaptureControl

WHSWorkLine::cancelLine

WmsArrivalCreateJournal::createWMSJournalTransFromTmp

WmsArrivalOverviewGeneration::updateOverviewInformation

WmsJournalCheckPostReception::initJournal

WMSOrderTrans::adjustQtyWMSOrderTrans

WMSOrderTrans::createNewWMSOrderTrans

WMSOrderTrans::insertOrUpdate

WMSOrderTrans::updateWMSOrderTrans

WmsPickingList_OrderPickDP.insertIntoTempTable, setWMSPickingList_OrderPickTmpTemplate
Enumerations made extensible

These enumerations have been made extensible in this update.

Additional extensibility enhancements

In addition to the refactored methods, the following extensibility enhancements have been made.

- Increase EDT string size for EcoResProductSearchName
- Change CacheLookup property to NotInTTS for AssetLedgerAccounts
- Change CacheLookup property to Found on TaxOnItem, TaxJurisdiction, TaxGroupData, and TaxData, and AssetLedgerAccounts
Refactored methods to support extensibility

These methods have been refactored to support extensibility through chain of command, delegates, or by providing access to members.

<table>
<thead>
<tr>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccDistProcessorProjectExtension.allocateExistingDistribution</td>
</tr>
<tr>
<td>AccDistProcessorProjectExtension.createDistributionLists</td>
</tr>
<tr>
<td>AccDistProcessorProjectExtension.ledgerDimensionAllocationList</td>
</tr>
<tr>
<td>AgreementConfirmationDP.processReport</td>
</tr>
<tr>
<td>Bank_FR.checkControlText</td>
</tr>
<tr>
<td>Bank_IT.checkCIN</td>
</tr>
<tr>
<td>Bank_IT.checkRegistrationNum</td>
</tr>
<tr>
<td>CustVendCheque.initTmpChequePrintout</td>
</tr>
<tr>
<td>FBSpedFileCreator_Contabil_BR.createRecordI052</td>
</tr>
<tr>
<td>HierarchyTemplateCopying_proj.createFromHierarchySource</td>
</tr>
<tr>
<td>HierarchyTreeLookup.datasource smmActivities.init</td>
</tr>
<tr>
<td>InventDim.validateFieldCombination</td>
</tr>
<tr>
<td>InventTransWMS_Register</td>
</tr>
<tr>
<td>InventUpd_Financial.updateFinancialIssue</td>
</tr>
<tr>
<td>InventUpd_Registered</td>
</tr>
<tr>
<td>JmgPostStandardSystem.PostProjTime</td>
</tr>
<tr>
<td>MarkupAllocation.run</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>MarkupTrans. MarkupTrans(datasource).active</td>
</tr>
<tr>
<td>PdsBatchAttribByItem.checkDuplicateAttributes</td>
</tr>
<tr>
<td>PdsBatchAttribByItem.validateFieldValue</td>
</tr>
<tr>
<td>PdsBatchAttribReserve.linkActive</td>
</tr>
<tr>
<td>PdsBatchAttributes.Datasource.PdsBatchAttributes.linkactive</td>
</tr>
<tr>
<td>ProjInvoiceProposalCreateLines.closeOK</td>
</tr>
<tr>
<td>ProjInvoiceProposalInsertLines.doCost</td>
</tr>
<tr>
<td>ProjInvoiceProposalInsertLines.doEmpl</td>
</tr>
<tr>
<td>ProjInvoiceProposalInsertLines.doItem</td>
</tr>
<tr>
<td>ProjInvoiceProposalInsertLines.doOnAccount</td>
</tr>
<tr>
<td>ProjPeriodPostingLedgerSales.enterItem</td>
</tr>
<tr>
<td>ProjPeriodPostingLedgerSales.enterCost</td>
</tr>
<tr>
<td>ProjPeriodPostingLedgerSales.enterEmpl</td>
</tr>
<tr>
<td>ProjPeriodPostingLedgerSales.enterRevenue</td>
</tr>
<tr>
<td>ProjPeriodPostingLedgerSales.run</td>
</tr>
<tr>
<td>ProjPlanVersionsManager.CopyHierarchy</td>
</tr>
<tr>
<td>ProjPlanVersionsManager::copyHierarchy</td>
</tr>
<tr>
<td>ProjPlanVersionsManager::createDraftVersion</td>
</tr>
<tr>
<td>ProjPlanVersionsManager::createTemplateHierarchy</td>
</tr>
<tr>
<td>PurchAutoCreate_PurchReq.initializeAndCreatePurchLine</td>
</tr>
<tr>
<td>PurchOrderLineSourceDocumentLineItem.calculateSourceDocumentAmountMap</td>
</tr>
<tr>
<td>PurchRFQPriceDiscAdmCreate.createPriceDiscAdmTrans</td>
</tr>
<tr>
<td>SalesFormLetter.validate</td>
</tr>
<tr>
<td>SalesTable2LineUpdatePrompt.salesTableFieldModifiedHandler</td>
</tr>
<tr>
<td>SalesUpdateRemain.cancelOpenOrderLinesDeliveryRemainder</td>
</tr>
</tbody>
</table>
**METHOD**

<table>
<thead>
<tr>
<th>WHSShipmentTable.createShipmentNotes</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHSUnShipLoadLineTmpDataCreator.createTmpLoadLineInventoryFromContainerLines</td>
</tr>
</tbody>
</table>

**Additional extensibility enhancements**

In addition to the refactored methods, the following extensibility enhancements have been made.

- Support for extensions to map: CustVendTrans
- Support for extensions to map: CustVendTransOpen
- Support extensibility for SQL statement: PriceDiscAdmCheckPost.postJournal
This is a list of extensibility features that were implemented in Dynamics 365 for Finance and Operations update 8.0.1. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Refactored methods to support extensibility

These methods have been refactored to support extensibility through chain of command, delegates, or by providing access to members.

- Class ProjControlPeriod::PeriodInsert
- Class ProjInvoiceChoose::doSalesLine
- Class CustInvoiceJour::printFreeTextJournal
- Class ProjCostControl.createEmptyTransactionType
- Class ProjEstimate::autoGenerateEstimateLinesFromTask
- Class ProjEstimateDataContract.updateEstimates
- Class ProjForecastBudget.Run
- Class ProjHierarchyProvider.preDeleteHierarchy
- Class ProjPlanVersionsManager::CopyTasks
- Class ProjPlanVersionsManager.createDraftFromPublishedVersion
- Class ProjPlanVersionsManager.PublishQuotationSubHierarchy
- Class ProjPlanVersionsManager.CreateDraftVersion
- Class ProjTask.addTask
- Class ProjInvoiceProposalCreateLines.performTransTypeSelectionCtrlLookup
- Class VendOpenTrans.editMarkTrans
- Class CustVendReversePosting.reverseTaxWithholdTrans
- Class CustVendSettle.postPennyDiff
<table>
<thead>
<tr>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class CustVendSettle.processStillOpenTransactions</td>
</tr>
<tr>
<td>Class InventTransferOrderOverviewDP.insertTmp</td>
</tr>
<tr>
<td>Class LedgerJournalTransCost.LedgerJournalTrans.Create</td>
</tr>
<tr>
<td>Class ProjAdjustmentSelect.dialog</td>
</tr>
<tr>
<td>Class ProjEstimate.syncEstimateLinesFromTask</td>
</tr>
<tr>
<td>Class ProjEstimateDataContract.UpdateEstimates</td>
</tr>
<tr>
<td>Class ProjForecastTransferFromWbs.transferToForecast</td>
</tr>
<tr>
<td>Class ProjWizardActivityCtrl.insertDBOnServer</td>
</tr>
<tr>
<td>Class ProjTaskEstimatesSynchronizer.calcTotalEstimateLineHours</td>
</tr>
<tr>
<td>Class ProjTaskEstimatesSynchronizer.countNumberOfHourEstimateLines</td>
</tr>
<tr>
<td>Class ProjTaskEstimatesSynchronizer.syncExistingHourEstimatesWithTask</td>
</tr>
<tr>
<td>Class ProjTaskEstimatesSynchronizer.syncHourEstimatesWithTaskEffort</td>
</tr>
<tr>
<td>Class ProjWbsCostPlanningServerActions.executeDataRetrievalAction</td>
</tr>
<tr>
<td>Class ProjWbsCostPlanningServerActions.getProjectCategoryTypes</td>
</tr>
<tr>
<td>Class ProjWbsSchedulePlanningServerActions.executeAction</td>
</tr>
<tr>
<td>Class ProjWbsSchedulePlanningServerActions.executeDataRetrievalAction</td>
</tr>
<tr>
<td>Class ProjStatisticCalc.validate</td>
</tr>
<tr>
<td>Class WHSInventReserve.insert</td>
</tr>
<tr>
<td>Class WHSInventReserveDelta.insert</td>
</tr>
<tr>
<td>Class ProjInvoiceProposalCreateLines.performTransTypeSelectionCtrlLookup</td>
</tr>
<tr>
<td>Class ProjJournalTransEmpl - Datasource: ProjJournalTrans.Validate</td>
</tr>
<tr>
<td>Class LedgerJournalEngine.initTaxItemGroup</td>
</tr>
<tr>
<td>Class LedgerJournalEngine.initValue</td>
</tr>
<tr>
<td>Class ProjJournalTrans.mergeResourceDimensionDefault</td>
</tr>
<tr>
<td>Class ProjTask.setTaskinfo</td>
</tr>
</tbody>
</table>
Other extensibility enhancements

In addition to the refactored methods, the following extensibility enhancements have been made.

- Support variable number of decimals - InventTestLowerLimit
- Support variable number of decimals - InventTestLowerTolerance
- Support variable number of decimals - InventTestStandardValue
- Support variable number of decimals - InventTestUpperLimit
- Support variable number of decimals - InventTestUpperTolerance
- Support to skip prompt on transaction reversal
- Enable extension of PSAProjQuotationApproval workflow
Hard-sealed application models

In Dynamics 365 for Finance and Operations version 8.0, all of Microsoft’s application models have been hard-sealed. Overlayered code in these models will now produce compilation errors. The only supported customization model is through extensions. If you cannot customize these models through extension, then you will have to make a request to Microsoft to enable extensibility by changing the standard application.

The following table includes a list of models that are now hard-sealed with this release.

<table>
<thead>
<tr>
<th>MODULE</th>
<th>MODEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplicationCommon</td>
<td>ApplicationCommon</td>
</tr>
<tr>
<td>ApplicationSuite</td>
<td>Electronic Reporting Application Suite Integration</td>
</tr>
<tr>
<td>ApplicationSuite</td>
<td>Foundation Upgrade</td>
</tr>
<tr>
<td>ApplicationSuite</td>
<td>Foundation</td>
</tr>
<tr>
<td>ApplicationSuite</td>
<td>SCMControls</td>
</tr>
<tr>
<td>ApplicationSuite</td>
<td>Tax Books Application Suite Integration</td>
</tr>
<tr>
<td>ApplicationSuite</td>
<td>Tax Engine Application Suite Integration</td>
</tr>
<tr>
<td>CaseManagement</td>
<td>CaseManagement</td>
</tr>
<tr>
<td>Currency</td>
<td>Currency</td>
</tr>
<tr>
<td>DataImpExpApplication</td>
<td>DataImpExpApplication</td>
</tr>
<tr>
<td>DataUpgrade</td>
<td>DataUpgrade</td>
</tr>
<tr>
<td>Directory</td>
<td>Directory</td>
</tr>
<tr>
<td>Directory</td>
<td>SecurityReports</td>
</tr>
<tr>
<td>GeneralLedger</td>
<td>GeneralLedger</td>
</tr>
<tr>
<td>Ledger</td>
<td>Ledger</td>
</tr>
<tr>
<td>PersonnelManagement</td>
<td>PersonnelManagement</td>
</tr>
</tbody>
</table>
Enumerations that have been made extensible

The following changes were made to support extending enumerations:

- Many enumerations in the standard application have been made extensible. An enumeration is made extensible by setting two properties on the enumeration. The `IsExtensible` property is set to `Yes`, and the `UseEnumValue` property is set to `No`.
- Some enumerations represent state. New façade methods have been added to help enable adding enumeration values by extension. For information about how to extend an enumeration, see Add values to enums through extension.
- Some application code that uses enumerations was changed to support extensibility. Common changes include:
  - Removing `throw` exception statements in the default case of a switch to allow post-event subscription.
  - Adding `SysExtension` support for extension.
  - Adding explicit delegates.

**ENUMERATION**

<table>
<thead>
<tr>
<th>BOMConsumeType</th>
<th>BOMFormula</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOMType</td>
<td></td>
</tr>
<tr>
<td>ChequeFormType</td>
<td></td>
</tr>
<tr>
<td>CostGroupType</td>
<td></td>
</tr>
<tr>
<td>CustAccountStatement</td>
<td></td>
</tr>
<tr>
<td>CustMandateScheme</td>
<td></td>
</tr>
<tr>
<td>CustVendDisputeStatus</td>
<td></td>
</tr>
<tr>
<td>DispositionAction</td>
<td></td>
</tr>
<tr>
<td>ItemCalcType</td>
<td></td>
</tr>
<tr>
<td>KMCollectionAnswerStatus</td>
<td></td>
</tr>
<tr>
<td>ENUMERATION</td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td></td>
</tr>
<tr>
<td>KanbanEventType</td>
<td></td>
</tr>
<tr>
<td>LedgerAccrualPeriod</td>
<td></td>
</tr>
<tr>
<td>LogisticsAddressElement</td>
<td></td>
</tr>
<tr>
<td>LogisticsLocationEntitytype</td>
<td></td>
</tr>
<tr>
<td>NoneBeginTransEnd</td>
<td></td>
</tr>
<tr>
<td>PSAInvoiceFormats</td>
<td></td>
</tr>
<tr>
<td>PdsCumulationPeriod</td>
<td></td>
</tr>
<tr>
<td>PdsRebateProgramType</td>
<td></td>
</tr>
<tr>
<td>PdsRebateTransaction</td>
<td></td>
</tr>
<tr>
<td>PdsUnitType</td>
<td></td>
</tr>
<tr>
<td>PriceDiscSystemSource</td>
<td></td>
</tr>
<tr>
<td>ProdFlushingPrincipBOM</td>
<td></td>
</tr>
<tr>
<td>ProdFlushingPrincipItem</td>
<td></td>
</tr>
<tr>
<td>ProdReservation</td>
<td></td>
</tr>
<tr>
<td>ProjAccountTypeCost</td>
<td></td>
</tr>
<tr>
<td>ProjAccountTypeSales</td>
<td></td>
</tr>
<tr>
<td>ProjAccountType</td>
<td></td>
</tr>
<tr>
<td>ProjJournalType</td>
<td></td>
</tr>
<tr>
<td>RevenueContributionMargin</td>
<td></td>
</tr>
<tr>
<td>SMATransactionType</td>
<td></td>
</tr>
<tr>
<td>SysPolicyRuleEnum</td>
<td></td>
</tr>
<tr>
<td>SysPolicyRuleTypeEnum</td>
<td></td>
</tr>
<tr>
<td>SysPolicyTypeEnum</td>
<td></td>
</tr>
<tr>
<td>TAMRebateAmtType</td>
<td></td>
</tr>
<tr>
<td>TAMVendRebateStatus</td>
<td></td>
</tr>
</tbody>
</table>
Data manipulation methods that do not raise DataEvents or missing insert, update, delete pre- and post-data events

As a general practice, you use data methods on tables to raise events that can be used for extending the application. The code base has not always followed this practice. For example, the `doInsert`, `doUpdate`, and `doDelete` data methods and certain table implementations did not make a call to `super()` in the data method.

The `insert`, `update`, and `delete` methods on the type classes have been refactored. Changes were made so that `super()` is called more consistently in data methods. These changes enable extensions to be added to these methods, so that pre- and post-events are now available for extension. The tables where the `insert`, `update`, and `delete` events were enabled for extension are listed in the following table.

<table>
<thead>
<tr>
<th>TYPE, NAME, DATA SOURCE, AND METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form ProjTableCreate.ProjTable.write</td>
</tr>
<tr>
<td>Form ReturnTable.ReturnTable.leaveRecord</td>
</tr>
<tr>
<td>Form SalesQuotationProjTable.SalesQuotationTable.leaveRecord</td>
</tr>
<tr>
<td>Form SalesQuotationTable.SalesQuotationTable.leaveRecord</td>
</tr>
<tr>
<td>Form SalesTable.SalesTable.leaveRecord</td>
</tr>
</tbody>
</table>

Refactored methods to support extensibility

These methods have been refactored to support extensibility through chain of command, delegates, or by providing access to members.

<table>
<thead>
<tr>
<th>TYPE, NAME, AND METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class AgreementConfirm_Sales.startConfirm</td>
</tr>
<tr>
<td>Class AssetChangeGroup.updateAssetGroupInfo</td>
</tr>
<tr>
<td>Class AssetPost.createAssetTransForPost</td>
</tr>
<tr>
<td>Class AssetSplit.getUpdatedSplitValueModel</td>
</tr>
<tr>
<td>Class AssetTableMethod.init</td>
</tr>
<tr>
<td>Class AssetTableMethod_SL.calc</td>
</tr>
<tr>
<td>TYPE, NAME, AND METHOD</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>Class AxSalesLine</td>
</tr>
<tr>
<td>Class BankPaymCancel.serverRun</td>
</tr>
<tr>
<td>Class BomSearch.New</td>
</tr>
<tr>
<td>Class BomSearch_BOMCopyType.New</td>
</tr>
<tr>
<td>Class Commission.run</td>
</tr>
<tr>
<td>Class CostSheetPanel.build</td>
</tr>
<tr>
<td>Class CreateInvoiceJournalPost.createFixedAsset</td>
</tr>
<tr>
<td>Class CustAccountStatementIntDPprintingAmountMST</td>
</tr>
<tr>
<td>Class CustCreditLimit.balanceEstimate</td>
</tr>
<tr>
<td>Class CustCreditLimit.calculateBalance</td>
</tr>
<tr>
<td>Class CustCreditLimit_SalesTable.New</td>
</tr>
<tr>
<td>Class CustInterestCreate</td>
</tr>
<tr>
<td>Class CustVoucher.post</td>
</tr>
<tr>
<td>Class DimensionDerivationRule.buildDimensionCombination</td>
</tr>
<tr>
<td>Class EcoResProductInformation.main</td>
</tr>
<tr>
<td>Class EcoResProductReleaseManager.setAndSaveRetailProductProperties</td>
</tr>
<tr>
<td>Class EcoResProductValidator.isEssentialFieldValuesSet</td>
</tr>
<tr>
<td>Class FormLetterServiceController.newFromContract</td>
</tr>
<tr>
<td>Class FormletterJournalPost.postLineDiscount</td>
</tr>
<tr>
<td>Class Graphics_WrkCtrCapBooking.insertLoad</td>
</tr>
<tr>
<td>Class Graphics_WrkCtrCapBooking.loadGroupReservations</td>
</tr>
<tr>
<td>Class Graphics_WrkCtrCapBooking.loadNumReservations</td>
</tr>
<tr>
<td>Class InterCompanyPostPurch.construct</td>
</tr>
<tr>
<td>Class InterCompanySyncPurchLineType</td>
</tr>
<tr>
<td>Class InterCompanySyncPurchTableType.setSalesTableData</td>
</tr>
<tr>
<td>Type, Name, and Method</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>Class InterCompanySyncPurchTableType</td>
</tr>
<tr>
<td>Class InventAgeDimDRcreateOrMergeInventAgeDimTmp</td>
</tr>
<tr>
<td>Class InventAgeDimDRinsertOrMergeInventAgeDimTmp</td>
</tr>
<tr>
<td>Class InventAgingCmdAggregateSelected.execute</td>
</tr>
<tr>
<td>Class InventCostItemDim.initInventSettlement</td>
</tr>
<tr>
<td>Class InventCostReport.newInventCostReport_CostBaseType</td>
</tr>
<tr>
<td>Class InventCountCreateItems.run</td>
</tr>
<tr>
<td>Class InventDimCtrl_Frm.clearInvisibleRanges</td>
</tr>
<tr>
<td>Class InventItemPriceActivationTaskActivateSim.activateOneInventItemPriceSim</td>
</tr>
<tr>
<td>Class InventItemPriceSim.moveSimulatedToCurrent</td>
</tr>
<tr>
<td>Class InventLedgerPostingDefinitionEntityHelper.inventAccountTypeX2InventAccountType</td>
</tr>
<tr>
<td>Class InventMov_SalesQuotation.isQuotationQtyEditable</td>
</tr>
<tr>
<td>Class InventProductDimensionLookup.dimEDT2FieldId</td>
</tr>
<tr>
<td>Class InventProductDimension</td>
</tr>
<tr>
<td>Class InventQualityManagementBlock.actOnAssociations</td>
</tr>
<tr>
<td>Class InventQualityManagementCreate.createOnRegistration</td>
</tr>
<tr>
<td>Class InventQualityManagementCreate.createQualityOrder</td>
</tr>
<tr>
<td>Class InventQualityManagementCreate.generateQualityOrders</td>
</tr>
<tr>
<td>Class InventQualityManagementCreateInvent.generateQualityOrdersWithDiscrimination</td>
</tr>
<tr>
<td>Class InventQualityMgmtCreateNonInvent.generateQualityOrdersWithDiscrimination</td>
</tr>
<tr>
<td>Class InventQualityOrderReopen.main</td>
</tr>
<tr>
<td>Class InventQualityOrderReopen.run</td>
</tr>
<tr>
<td>Class InventQualityOrderValidate.main</td>
</tr>
<tr>
<td>Class InventQualityOrderValidate.run</td>
</tr>
<tr>
<td>Class InventQualityReferenceTypeSales.isEligibleForQualityManagement</td>
</tr>
<tr>
<td>TYPE, NAME, AND METHOD</td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>Class InventQualityReferenceTypeSales.supportsInventoryBlocking</td>
</tr>
<tr>
<td>Class InventQualitymanagementCreate.createPerQualityAssociations</td>
</tr>
<tr>
<td>Class InventSumReCalcItem.updateActualInventSum</td>
</tr>
<tr>
<td>Class InventTestAssociationTable.checkAccountRelation</td>
</tr>
<tr>
<td>Class InventTestAssociationTable.initRecord</td>
</tr>
<tr>
<td>Class InventTrackingDimTracingCriteria.initFromArgs</td>
</tr>
<tr>
<td>Class InventTransLine.insert</td>
</tr>
<tr>
<td>Class InventTransferMulti.run</td>
</tr>
<tr>
<td>Class InventTransferMultiReceiver::main</td>
</tr>
<tr>
<td>Class InventTransferMultiShip.buildParmFromWMSShipment</td>
</tr>
<tr>
<td>Class InventTransferMultiShip.runUpdate</td>
</tr>
<tr>
<td>Class InventTransferOrderOverviewDP.insertTmpTable</td>
</tr>
<tr>
<td>Class InventTransferUpdShip.updateInventTransferLine</td>
</tr>
<tr>
<td>Class InventUpd_Physical.updatePhysicalIssue</td>
</tr>
<tr>
<td>Class InventUpd_Physical.updatePhysicalReturnedReceipt</td>
</tr>
<tr>
<td>Class InventUpd_Picked.updatePickInventTrans</td>
</tr>
<tr>
<td>Class InventUpd_Reservation.whsUpdateReserveMore</td>
</tr>
<tr>
<td>Class InventUpdate.raiseOnHandChangingOnPhysicalStatusUpd</td>
</tr>
<tr>
<td>Class InventUpdate.updateDimReservePhysical</td>
</tr>
<tr>
<td>Class InventUpdate.updateTransDimTransferReceipt</td>
</tr>
<tr>
<td>Class InventUpdate.writeInventTransAutoDim</td>
</tr>
<tr>
<td>Class InventValueReportDP.processInventValueReportTmpLine</td>
</tr>
<tr>
<td>Class InventoryMainAccDimensionListProvider.populateMainAccountDimensionList</td>
</tr>
<tr>
<td>Class LedgerBalanceQueryGeneralJournal.addToBalanceTotals</td>
</tr>
<tr>
<td>Class LedgerBalanceQueryGeneralJournal.createQuery</td>
</tr>
<tr>
<td>Class</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>LedgerJournalCheckPost</td>
</tr>
<tr>
<td>LedgerJournalCheckPost</td>
</tr>
<tr>
<td>LedgerJournalCheckPost</td>
</tr>
<tr>
<td>LedgerJournalCheckPost</td>
</tr>
<tr>
<td>LedgerJournalGetTransactions</td>
</tr>
<tr>
<td>LedgerVoucherObject</td>
</tr>
<tr>
<td>LedgerVoucherTransObject</td>
</tr>
<tr>
<td>LogisticsLocationSelectForm</td>
</tr>
<tr>
<td>LogisticsLocationSelectForm</td>
</tr>
<tr>
<td>LogisticsPostalAddressFormHandlerExt</td>
</tr>
<tr>
<td>MCRItemListGeneration</td>
</tr>
<tr>
<td>MCRItemListGeneration</td>
</tr>
<tr>
<td>MCRMarginAlert</td>
</tr>
<tr>
<td>Markup</td>
</tr>
<tr>
<td>MarkupAllocationSelectionManager</td>
</tr>
<tr>
<td>PSAProjInvoiceDP</td>
</tr>
<tr>
<td>PSAProjInvoiceDP</td>
</tr>
<tr>
<td>PdsApprovedVendorListCheck</td>
</tr>
<tr>
<td>PlanActivityTimeCalculation</td>
</tr>
<tr>
<td>PordJournalCreateBOM</td>
</tr>
<tr>
<td>PriceDisc</td>
</tr>
<tr>
<td>PriceDisc</td>
</tr>
<tr>
<td>PriceDisc</td>
</tr>
<tr>
<td>PriceDisc</td>
</tr>
<tr>
<td>PriceTypeConverter</td>
</tr>
<tr>
<td>TYPE, NAME, AND METHOD</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>Class PrintMgmtReportFormatSubscriber.add</td>
</tr>
<tr>
<td>Class PrintMgmtReportFormatSubscriber.populate</td>
</tr>
<tr>
<td>Class ProdBOM.prodFlushingPrincipItem2BOM</td>
</tr>
<tr>
<td>Class ProdJournalCreateBOM.createLinesInventTrans</td>
</tr>
<tr>
<td>Class ProdJournalCreateBOM.createLinesInventTrans</td>
</tr>
<tr>
<td>Class ProdJournalCreateBOM.createLinesProdBOM</td>
</tr>
<tr>
<td>Class ProdJournalCreateBOM.dialog</td>
</tr>
<tr>
<td>Class ProdJournalCreateBOM.validate</td>
</tr>
<tr>
<td>Class ProdJournalFormTransBOM.setupCWFomControl</td>
</tr>
<tr>
<td>Class ProdPickListController.prePromptModifyContract</td>
</tr>
<tr>
<td>Class ProdPicklistDPInterValues</td>
</tr>
<tr>
<td>Class ProdStatusType_Released.checkPostJournal</td>
</tr>
<tr>
<td>Class ProdTableListPageInteraction.getEnabledControls</td>
</tr>
<tr>
<td>Class ProdUpdReportFinished.updateBomConsumption</td>
</tr>
<tr>
<td>Class ProdUpdReportFinished.updateRouteConsumption</td>
</tr>
<tr>
<td>Class ProdUpdSplit.createSplitToProduction</td>
</tr>
<tr>
<td>Class ProdUpdStartUp.getListOfBOMJournals</td>
</tr>
<tr>
<td>Class ProdUpdStartUp.updateBOMConsumption</td>
</tr>
<tr>
<td>Class ProjInvoiceDPInterIntoProjInvoiceTmp</td>
</tr>
<tr>
<td>Class ProjInvoiceProposalInsertLines.run</td>
</tr>
<tr>
<td>Class ProjInvoiceProposalInsertLines::run()</td>
</tr>
<tr>
<td>Class ProjPlanVersionsManager</td>
</tr>
<tr>
<td>Class ProjPostItemJournal::projTransCreate</td>
</tr>
<tr>
<td>Class ProjProposalTotals.calc</td>
</tr>
<tr>
<td>Class PsaProjInvoiceDPInterProformaPSAProjInvoiceTmp</td>
</tr>
<tr>
<td>TYPE, NAME, AND METHOD</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>Class PsacustomerRetention.createFeeTransactionForProposal</td>
</tr>
<tr>
<td>Class PurchAgreementGenerateReleaseOrder.check</td>
</tr>
<tr>
<td>Class PurchAgreementGenerateReleaseOrder.validatePurchLinesWithPurchQty</td>
</tr>
<tr>
<td>Class PurchAutoCreate.construct</td>
</tr>
<tr>
<td>Class PurchAutoCreate.construct</td>
</tr>
<tr>
<td>Class PurchAutoCreate_RFQ.construct</td>
</tr>
<tr>
<td>Class PurchAutoCreate_SalesProjectItemReq.createLine</td>
</tr>
<tr>
<td>Class PurchAutoCreate_SalesProjectItemReq.createPurchLine</td>
</tr>
<tr>
<td>Class PurchCancel.parmPurchTable</td>
</tr>
<tr>
<td>Class PurchCancel.run</td>
</tr>
<tr>
<td>Class PurchCopying.deleteLines</td>
</tr>
<tr>
<td>Class PurchCreateFromSalesOrder</td>
</tr>
<tr>
<td>Class PurchFormLetterParmData.createParmLine</td>
</tr>
<tr>
<td>Class PurchFormLetterParmDataInvoice.createParmLineAndSubLines</td>
</tr>
<tr>
<td>Class PurchFormletterParmData.reSelectLines</td>
</tr>
<tr>
<td>Class PurchFormletterParmDataApproveJournal.updateQueryBuild</td>
</tr>
<tr>
<td>Class PurchFormletterParmDataInvoice</td>
</tr>
<tr>
<td>Class PurchInvoiceCreate.createJournalLine</td>
</tr>
<tr>
<td>Class PurchInvoiceJournalCreate.checkInvoicePolicies</td>
</tr>
<tr>
<td>Class PurchInvoiceJournalCreate.checkMatching</td>
</tr>
<tr>
<td>Class PurchInvoiceJournalPost.createFixedAsset</td>
</tr>
<tr>
<td>Class PurchInvoiceJournalPost.lateMatchPackingSlip</td>
</tr>
<tr>
<td>Class PurchLineType.validateWrite</td>
</tr>
<tr>
<td>Class PurchLineVersioningFieldSet.isChangeConfirmationRequired</td>
</tr>
<tr>
<td>Class PurchOrderLineSourceDocumentLineItem.calculateSourceDocumentAmountMap</td>
</tr>
<tr>
<td>Type, Name, and Method</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>Class PurchOrderLineSourceDocumentLineItem.calculateSourceDocumentAmountMap</td>
</tr>
<tr>
<td>Class PurchOrderLineSourceDocumentLineItem.calculateSourceDocumentAmountMap</td>
</tr>
<tr>
<td>Class PurchPackingSlipDP.createProductReceiptLines</td>
</tr>
<tr>
<td>Class PurchPackingSlipJournalPost.selectFormletterJournalTrans</td>
</tr>
<tr>
<td>Class PurchRFQCaseAutoCreate.newAutoCreate</td>
</tr>
<tr>
<td>Class PurchReApprovalPolicyRuleFieldList.addTable2Hierarchy</td>
</tr>
<tr>
<td>Class PurchReApprovalPolicyRuleFieldList.addTable2Hierarchy</td>
</tr>
<tr>
<td>Class PurchSelectLinesManager.passSets</td>
</tr>
<tr>
<td>Class PurchTableInteraction.enableHeaderPurchase</td>
</tr>
<tr>
<td>Class PurchTableInteractionHelper.getJournalEnquiryButtons</td>
</tr>
<tr>
<td>Class PurchTableInteractionHelper.getUpdateJournalButtons</td>
</tr>
<tr>
<td>Class PurchaseOrderResponseConsume.checkIfPurchLinesRequireUpdate</td>
</tr>
<tr>
<td>Class PurchaseOrderResponseConsume.checkIfResponseLineCannotBeConsumedAndUpdateConsumptionState</td>
</tr>
<tr>
<td>Class PurchaseOrderResponseConsume.consumeFirstPurchaseOrderResponseLineAndInitiateArchivingOnPurchaseLine</td>
</tr>
<tr>
<td>Class PurchaseOrderResponseConsume.consumeRemainingPurchaseOrderResponseLines</td>
</tr>
<tr>
<td>Class PurchaseOrderResponseConsumeLine.checkIfSelectedPurchLinesRequireUpdate</td>
</tr>
<tr>
<td>Class ReqCalc.covCodeQty</td>
</tr>
<tr>
<td>Class ReqCalc.insertItemInventSum</td>
</tr>
<tr>
<td>Class ReqCalc.insertItemInventTrans</td>
</tr>
<tr>
<td>Class ReqTransFormPo.validateFromInventLocationId</td>
</tr>
<tr>
<td>Class ReqTransPoMarkChangeToRFQ.DialogPostRun</td>
</tr>
<tr>
<td>Class ReqTransPoMarkFirm.createPurchLine</td>
</tr>
<tr>
<td>Class ReqTransPoMarkFirm.setPurchTable</td>
</tr>
<tr>
<td>Class RetailAssortmentLookupTask.explodeAssortments</td>
</tr>
<tr>
<td>Class RetailCreateLinesFromProductsToAdd.createPeriodicDiscount</td>
</tr>
<tr>
<td>TYPE, NAME, AND METHOD</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>Class RetailCreateLinesFromProductsToAdd.loadLines</td>
</tr>
<tr>
<td>Class RetailPackagePurchManagement.createLines</td>
</tr>
<tr>
<td>Class RetailProductPropertyManager.saveInventTableAndRelated</td>
</tr>
<tr>
<td>Class RetailProductPropertyManager.validateWriteOnInventTable</td>
</tr>
<tr>
<td>Class RetailSalesOrderCalculator.saveSalesOrder</td>
</tr>
<tr>
<td>Class RetailSalesOrderCalculator.setPriceOnCurrentLine</td>
</tr>
<tr>
<td>Class RetailSalesQuotationCalculator.saveSalesQuote</td>
</tr>
<tr>
<td>Class RetailSalesQuotationCalculator.setPricesOnCurrentLine</td>
</tr>
<tr>
<td>Class ReturnTableInteraction.enableControl</td>
</tr>
<tr>
<td>Class RouteCopyToRoute.insertRouteOpr</td>
</tr>
<tr>
<td>Class SMAServiceFunctionLine_Transfer.checkJournalType</td>
</tr>
<tr>
<td>Class SMAServiceFunctionLine_Transfer.postJournalType</td>
</tr>
<tr>
<td>Class SMAServiceFunctionLine_Transfer.sumjournals</td>
</tr>
<tr>
<td>Class SMAServiceOrderCreate.createServiceOrderLine</td>
</tr>
<tr>
<td>Class SalesAutoCreate_ReleaseFromAgreement.createSalesTable</td>
</tr>
<tr>
<td>Class SalesCancelOrder.run</td>
</tr>
<tr>
<td>Class SalesCopying.copy</td>
</tr>
<tr>
<td>Class SalesCreateOrderFromCustomer.main</td>
</tr>
<tr>
<td>Class SalesFormLetter.mainOnServer</td>
</tr>
<tr>
<td>Class SalesFormLetterParmData.createParmLine</td>
</tr>
<tr>
<td>Class SalesFormLetterReport.construct</td>
</tr>
<tr>
<td>Class SalesFormLetterParmData.reSelectLines</td>
</tr>
<tr>
<td>Class SalesFormLetterParmDataInvoice.reSelectInit</td>
</tr>
<tr>
<td>Class SalesInvoiceDP.invoiceTxt</td>
</tr>
<tr>
<td>Class SalesInvoiceDP.itemId</td>
</tr>
<tr>
<td>TYPE, NAME, AND METHOD</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>Class SalesLineCopyFromSource.updateCopiedLine</td>
</tr>
<tr>
<td>Class SalesLineType.setReservation</td>
</tr>
<tr>
<td>Class SalesLineType.setSalesStatus</td>
</tr>
<tr>
<td>Class SalesLineType.syncPurchLine</td>
</tr>
<tr>
<td>Class SalesPackingSlipDP.createSalesPackingSlipLines</td>
</tr>
<tr>
<td>Class SalesPackingSlipJournalPost.addToInventReportDimHistory</td>
</tr>
<tr>
<td>Class SalesPurchLineInterface.setPriceAgreement</td>
</tr>
<tr>
<td>Class SalesQuotationCopying.copyHeader</td>
</tr>
<tr>
<td>Class SalesQuotationEditLinesForm_Sales_Confir.createSalesTable</td>
</tr>
<tr>
<td>Class SalesQuotationEditLinesForm</td>
</tr>
<tr>
<td>Class SalesQuotationLineType_Sales.validateWrite</td>
</tr>
<tr>
<td>Class SalesTableListPageInteraction.setButtonInterCompany</td>
</tr>
<tr>
<td>Class SalesTableType.checkUpdate</td>
</tr>
<tr>
<td>Class SalesTableType.interCompanyMirror</td>
</tr>
<tr>
<td>Class SmmCampaignQueries</td>
</tr>
<tr>
<td>Class SmmLeadUpdate</td>
</tr>
<tr>
<td>Class SmmOpportunityLink</td>
</tr>
<tr>
<td>Class SmmUpdateBusRel.updateFromCustTableSFA2</td>
</tr>
<tr>
<td>Class TradeCurrencyConversionPrompt.construct</td>
</tr>
<tr>
<td>Class TradeLineRenumbering.renumber</td>
</tr>
<tr>
<td>Class TradeTotals.calc</td>
</tr>
<tr>
<td>Class VendDocumentLineInterface.setPurchaseQty</td>
</tr>
<tr>
<td>Class VendInvoicePolicyValidation.policyViolationMessage</td>
</tr>
<tr>
<td>Class VendProvisionalBalanceDP.processReport</td>
</tr>
<tr>
<td>Class WHSPool.pickFromWorkCenter</td>
</tr>
<tr>
<td>TYPE, NAME, AND METHOD</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>Class WHSShipConfirm.createUOMStructure</td>
</tr>
<tr>
<td>Class WHSWorkExecute.pickLicensePlateHandledByLP</td>
</tr>
<tr>
<td>Class WhsInventOnHandReserve.changeReservation</td>
</tr>
<tr>
<td>Class WhsInventOnHandReserve.setMovement</td>
</tr>
<tr>
<td>Class WhsPackForm.buttonPack_clicked</td>
</tr>
<tr>
<td>Class WhsPostEngineBase.createLoadFromShipment</td>
</tr>
<tr>
<td>Class WhsShipConfirm.createInventTransferParmLineTMS</td>
</tr>
<tr>
<td>Class WhsShipConfirm.createInventTransferParmLine</td>
</tr>
<tr>
<td>Class WhsWorkExecute</td>
</tr>
<tr>
<td>Class WmsBillOfLadingDP::insertIntoTempTable</td>
</tr>
<tr>
<td>Class WmsOrderTransType_OutputDontPostTransfer.updateParentMovement</td>
</tr>
<tr>
<td>Class WrkCtrCapResHandler.hasNewCapacityReservation</td>
</tr>
<tr>
<td>Class WrkCtrCapResHandler.loadCapacityReservations</td>
</tr>
<tr>
<td>Class WrkCtrReservedSum.calcReservationSumGroupId</td>
</tr>
<tr>
<td>Class WrkCtrReservedSum.calcReservationSumGroupId</td>
</tr>
<tr>
<td>Class WrkCtrReservedSum.calcReservationSumWrkCtrId</td>
</tr>
<tr>
<td>Class WrkCtrReservedSum.calcReservationSumWrkCtrId</td>
</tr>
<tr>
<td>Class WrkCtrScheduler_Prod.saveOperation</td>
</tr>
<tr>
<td>Class createParmLinesFromTransferLinesOnLoad</td>
</tr>
<tr>
<td>Class smmCampaignQueries.lookupClass</td>
</tr>
<tr>
<td>Entity EcoResProductDimensionGroupEntity.dataSourceDimensionFieldId</td>
</tr>
<tr>
<td>Entity InventProductDefaultOrderSettingsEntity.insertEntityDataSource</td>
</tr>
<tr>
<td>Entity InventProductSiteSpecificOrderSettingsEntity.insertEntityDataSource</td>
</tr>
<tr>
<td>Entity PSAActualEntity.createQuery_LaborConsumptionQty</td>
</tr>
<tr>
<td>Entity PSAActualEntity.createQuery_LaborConsumption</td>
</tr>
<tr>
<td>TYPE, NAME, AND METHOD</td>
</tr>
<tr>
<td>----------------------------------------</td>
</tr>
<tr>
<td>Entity PSAActualEntity.createQuery_PILaborCost</td>
</tr>
<tr>
<td>Entity PSAActualEntity.createQuery_PILaborQty</td>
</tr>
<tr>
<td>Entity PSAForecastEntity.createQuery_LaborConsumptionForecastQty</td>
</tr>
<tr>
<td>Entity PSAForecastEntity.createQuery_LaborConsumptionForecast</td>
</tr>
<tr>
<td>Entity PSAForecastEntity.createQuery_PILaborForecastCost</td>
</tr>
<tr>
<td>Entity PSAForecastEntity.createQuery_PILaborForecastQty</td>
</tr>
<tr>
<td>Form BOMCalcDialog.updateDesign</td>
</tr>
<tr>
<td>Form EcoResProductCreate.releaseProductToCompany</td>
</tr>
<tr>
<td>Form InventItemOrderSetup.InventItemSetupSupplyType.editOrderType</td>
</tr>
<tr>
<td>Form InventLocationIdLookup.InventDim_DS.init</td>
</tr>
<tr>
<td>Form InventLocationIdLookup.InventLocation_DS.init</td>
</tr>
<tr>
<td>Form InventNonConformanceTable.init</td>
</tr>
<tr>
<td>Form InventNonConformanceTableCreate.InventNonConformanceTable.write</td>
</tr>
<tr>
<td>Form InventQualityOrderTableCreate.allowEdit</td>
</tr>
<tr>
<td>Form InventQualityOrderTableCreate.refreshCaller</td>
</tr>
<tr>
<td>Form InventTestAssociationTable.initRecord</td>
</tr>
<tr>
<td>Form InventTransPick\ImplInventTransWMS.validateWrite</td>
</tr>
<tr>
<td>Form LedgerTransVoucher.updateQueryForProject</td>
</tr>
<tr>
<td>Form MarkupAllocation.init</td>
</tr>
<tr>
<td>Form MarkupAllocation_VendInvoiceTrans</td>
</tr>
<tr>
<td>Form PdsBatchAttributes.PdsBatchAttributes.linkActive</td>
</tr>
<tr>
<td>Form PriceDiscAdmTable.init</td>
</tr>
<tr>
<td>Form PriceDiscTable.appendInventCriteria</td>
</tr>
<tr>
<td>Form PriceDiscTable.buildOrderLineFilter</td>
</tr>
<tr>
<td>Form PriceDiscTable.buildSearchFilter</td>
</tr>
<tr>
<td>TYPE, NAME, AND METHOD</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>Form PriceDiscTable.isLineFilterEnabled</td>
</tr>
<tr>
<td>Form PriceDiscTable.retrieveRelationType</td>
</tr>
<tr>
<td>Form ProdParmStartUp.ProdParmStartUp.active</td>
</tr>
<tr>
<td>Form ProjCreditNoteSelect.editMark</td>
</tr>
<tr>
<td>Form ProjTableCreate.ProjTable.write</td>
</tr>
<tr>
<td>Form PurchCreateFromSalesOrder.initFields</td>
</tr>
<tr>
<td>Form PurchUpdateRemain.closeOk</td>
</tr>
<tr>
<td>Form ReqTransPoMarkFirm.init</td>
</tr>
<tr>
<td>Form RetailAddItems.closeOk</td>
</tr>
<tr>
<td>Form RetailColorGroupTable.RetailColorGroupTrans.recordHasChanges</td>
</tr>
<tr>
<td>Form RouteLookupOprNum.init</td>
</tr>
<tr>
<td>Form VendEditInvoice.invoiceAccountModified</td>
</tr>
<tr>
<td>Form VendEditInvoice.run</td>
</tr>
<tr>
<td>Form VendOpenTrans.editMarkTrans</td>
</tr>
<tr>
<td>Form WrkCtrCapResGraphDialog.setParm</td>
</tr>
<tr>
<td>Map BomCalcTransMap.displayUnitId</td>
</tr>
<tr>
<td>Table AssetTable.lookupAccountNum</td>
</tr>
<tr>
<td>Table AssetTrans</td>
</tr>
<tr>
<td>Table CaseDetailBase.validateWrite</td>
</tr>
<tr>
<td>Table EcoResProductMasterConfiguration.existWithSameConfigUnit</td>
</tr>
<tr>
<td>Table FormletterJournalTrans.getLinePrefix</td>
</tr>
<tr>
<td>Table InventItemPriceSim.autoSalesPrice</td>
</tr>
<tr>
<td>Table InventQualityOrderLine.adjustInt</td>
</tr>
<tr>
<td>Table InventQualityOrderTable.createInventQualityOrderLines</td>
</tr>
<tr>
<td>Table InventQualityOrderTable.initFromReference</td>
</tr>
<tr>
<td>Type, Name, and Method</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>Table InventQualityOrderTable.initQtyFromAssociation</td>
</tr>
<tr>
<td>Table InventTestAssociationTable.checkAccountRelation</td>
</tr>
<tr>
<td>Table InventTestAssociationTable.validateWrite</td>
</tr>
<tr>
<td>Table InventTrans.insertReturnTransOrigin</td>
</tr>
<tr>
<td>Table InventTransOrigin.createOrigin</td>
</tr>
<tr>
<td>Table InventTransferParmLine.createPickLines</td>
</tr>
<tr>
<td>Table InventTransferParmLine.createReceiveLines</td>
</tr>
<tr>
<td>Table InventTransferParmLine.createShipLines</td>
</tr>
<tr>
<td>Table JmgStampJournalTrans</td>
</tr>
<tr>
<td>Table JmgTermReg.createJournalSignIn</td>
</tr>
<tr>
<td>Table JmgTermReg.update</td>
</tr>
<tr>
<td>Table LedgerJournalTrans.delete</td>
</tr>
<tr>
<td>Table LedgerJournalTrans.validateWrite_Server</td>
</tr>
<tr>
<td>Table PriceDiscAdm.getEntityAutoReportFieldGroupName</td>
</tr>
<tr>
<td>Table PriceDiscAdm.getEntityJournalNumberFieldName</td>
</tr>
<tr>
<td>Table PriceDiscAdmTrans.CheckAccountRelation</td>
</tr>
<tr>
<td>Table PriceDiscAdmTrans.checkItemRelation</td>
</tr>
<tr>
<td>Table PurchLine.setPriceDisc</td>
</tr>
<tr>
<td>Table RouteVersion.selectRouteVersion</td>
</tr>
<tr>
<td>Table SalesLine.checkItemId</td>
</tr>
<tr>
<td>Table SalesLine.getSourcingFields</td>
</tr>
<tr>
<td>Table SalesLine.setPriceAgreement</td>
</tr>
<tr>
<td>Table SalesLine.setPriceDisc</td>
</tr>
<tr>
<td>Table SalesLine.setSourcingFields</td>
</tr>
<tr>
<td>Table SalesQuotationLine.IsCategoryBased</td>
</tr>
</tbody>
</table>
Maps enabled for extensibility

New patterns have been introduced for maps implementation that will allow you to add fields and methods by extensions. Details on how this is done is available in the documentation both with maps that are used as interfaces and for versioning implementations.

The following table lists the maps and related tables where changes have been applied for enabling extensibility.

<table>
<thead>
<tr>
<th>TYPE, NAME, AND METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table SalesQuotationLine.mcrCreateFromTmpFrmVirtualFromContract</td>
</tr>
<tr>
<td>Table SalesQuotationLine.setPriceAgreement</td>
</tr>
<tr>
<td>Table SalesQuotationLine.setPriceDisc</td>
</tr>
<tr>
<td>Table SalesLine.splitReturnLine</td>
</tr>
<tr>
<td>Table SupplItemCreate.createLine</td>
</tr>
<tr>
<td>Table TmpInventTransMark.packTmpMark</td>
</tr>
<tr>
<td>Table VendInvoiceMatchingLine.initFromPurchLine</td>
</tr>
<tr>
<td>Table VendTable.updateOnHold</td>
</tr>
<tr>
<td>Table WHSInvent.checkNonPhysicalDims</td>
</tr>
<tr>
<td>Table WHSShipmentTable.consolidateShipments</td>
</tr>
<tr>
<td>Table WHSShipmentTable.transferShipment</td>
</tr>
<tr>
<td>Table WHSTmpPackingLine.addTmpPackLine</td>
</tr>
<tr>
<td>Table salesLine.initFromProjTable</td>
</tr>
<tr>
<td>Table smmBusRefTable.updateReferences</td>
</tr>
<tr>
<td>Table smmLeadTable</td>
</tr>
<tr>
<td>Table smmOpportunityTable</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MAPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustVendSettlement</td>
</tr>
<tr>
<td>JmgStampTransMap</td>
</tr>
<tr>
<td>PriceDiscResultFields</td>
</tr>
<tr>
<td>SalesPurchLine</td>
</tr>
</tbody>
</table>
Inventory dimensions

This release made minor improvements to the new model for adding inventory dimensions, all targeted at supporting more scenarios through extensions.

**CHANGE**

- BOM hierarchy works only with the config dimension
- Form BOMDesigner should use field group for showing dimensions
- Form EcoResProductSearchLookup should use field group for showing dimensions
- Form FactureJournal_RU should use field group for showing dimensions
- Form InventDimParmFixed.InventDimensionXXFlag.Style is incorrect
- Form InventItemOrderSetup should use field group for showing dimensions
- Form InventTransferParmPick should use field group for showing dimensions
- Form InventTransferReleaseOrderPicking should use field group for showing dimensions
- Form KanbanCreateScheduled should use field group for showing dimensions
- Form KanbanJobPickingListPart should use field group for showing dimensions
- Form KanbanRules should use field group for showing dimensions
- Form LeanPeggingTree should use field group for showing dimensions
- Form MCRItemDisplay should use field group for showing dimensions
- Form MCRPriceDiscGroupItem should use field group for showing dimensions
- Form PlanActivityServiceWizard should use field group for showing dimensions
- Form ProdBOMVendor should use field group for showing dimensions
- Form PurchAgreementGenerateReleaseOrder should use field group for showing dimensions
- Form PurchAgreementHistory should use field group for showing dimensions
- Form PurchComplementaryInvoice should use field group for showing dimensions
- Form PurchRFQCompareLineDimensions should use field group for showing dimensions
- Form PurchTable.TrackingDimesions has incorrect spelling
- Form PurchVendorPortalAllResponse should use field group for showing dimensions
- Form PurchVendorPortalConfirmedOrders should use field group for showing dimensions
<table>
<thead>
<tr>
<th>Form / Table Name</th>
<th>Format Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>PurchVendorPortalOriginalOrder</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>PurchVendorPortalRequests</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>PurchVendorPortalResponses</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>ReqDemPlanEasyItemAllocator</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>ReqOutboundIntercompanyDemand</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>ReqSupplyDemandScheduleFilters</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>RetailVariantLookup</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>RouteVersionFeasibility</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>SMAAgreementTable</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>SalesAgreementGenerateReleaseOrder</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>SalesAgreementHistory</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>SalesComplementaryInvoice</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>SalesLineDeliveryDetails</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>SalesQuotationProjTable</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>SalesQuotationTable</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>SalesTable</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>TAMFundManagement</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>TAMTradePromotions</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>VendEditInvoice</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>VendJournalMatch_PackingSlip</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>WHSLoadPlanningWorkBench</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>WHSLoadPlanningWorkbench</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>WHSLoadTable</td>
<td>Use field group for showing dimensions</td>
</tr>
<tr>
<td>WHSProdWaveTableManageBOMPool</td>
<td>Use field group for showing dimensions</td>
</tr>
</tbody>
</table>
Form WHSWorkTable should use field group for showing dimensions

Form WMSOrderTransUnPick should use field group for showing dimensions

Form WMSPickingRegistration should use field group for showing dimensions

Report InventAging does not support extra dimensions

Table EcoResProductVariantStaging.StagingIdx need extra dimension fields

Other changes
The following table lists additional changes that have been made for extensibility.

<table>
<thead>
<tr>
<th>CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add filter interface: form InventQualityOrderTable</td>
</tr>
<tr>
<td>Address management: Adding new address fields</td>
</tr>
<tr>
<td>AxMaps - TradePostalAddress - partyTable</td>
</tr>
<tr>
<td>Bank Trans Comments - BankReconciliationDataInitializer</td>
</tr>
<tr>
<td>Cancellation Log Requirements - Update Sales Deliver Remainder</td>
</tr>
<tr>
<td>Extend the grouping mechanisme from purch req line to purch line</td>
</tr>
<tr>
<td>Extend the splitting mechanisme from purch req line to purch line</td>
</tr>
<tr>
<td>Allow multiple funding sources in conjunction with item requirements</td>
</tr>
<tr>
<td>Implementing exchange rate provider framework</td>
</tr>
<tr>
<td>Make the PriceDiscPartyCodeType extensible in all usages</td>
</tr>
<tr>
<td>Make the PriceDiscProductCodeType extensible in all usages</td>
</tr>
<tr>
<td>Table RetailChannelTable does not have ReplacementKey</td>
</tr>
<tr>
<td>Table RetailSeasonTable CreateRecIdIndex True</td>
</tr>
<tr>
<td>Modify index: table InventTestAssociationTable</td>
</tr>
<tr>
<td>Entity UnitOfMeasureEntity switched to public</td>
</tr>
<tr>
<td>Entity UnitOfMeasureTranslationEntity switched to public</td>
</tr>
</tbody>
</table>
This topic lists the extensibility features that were released in Dynamics 365 for Finance and Operations, Enterprise edition 7.3. For more information about the schedule of changes that support extensibility, see Application extensibility plans.

Soft-sealed application models

This release marks the last release before all models will become hard-sealed, and as a step toward this all application models are now soft-sealed. Soft-sealed models still allow for making overlayered code, but warnings will be generated when you compile the overlayered code.

**NOTE**

You can still overlayer code, but extension is the recommended approach.

The following table includes a list of the models that are soft-sealed with this release.

<table>
<thead>
<tr>
<th>MODULE</th>
<th>MODEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplicationCommon</td>
<td>ApplicationCommon</td>
</tr>
<tr>
<td>ApplicationSuite</td>
<td>Electronic Reporting Application Suite Integration</td>
</tr>
<tr>
<td>ApplicationSuite</td>
<td>Foundation Upgrade</td>
</tr>
<tr>
<td>ApplicationSuite</td>
<td>Foundation</td>
</tr>
<tr>
<td>ApplicationSuite</td>
<td>SCMControls</td>
</tr>
<tr>
<td>ApplicationSuite</td>
<td>Tax Books Application Suite Integration</td>
</tr>
<tr>
<td>ApplicationSuite</td>
<td>Tax Engine Application Suite Integration</td>
</tr>
<tr>
<td>CaseManagement</td>
<td>CaseManagement</td>
</tr>
<tr>
<td>Currency</td>
<td>Currency</td>
</tr>
<tr>
<td>DataImpExpApplication</td>
<td>DataImpExpApplication</td>
</tr>
<tr>
<td>DataUpgrade</td>
<td>DataUpgrade</td>
</tr>
<tr>
<td>Directory</td>
<td>Directory</td>
</tr>
<tr>
<td>Directory</td>
<td>SecurityReports</td>
</tr>
</tbody>
</table>
With this release, almost all application core models have been hard-sealed. Overlayered code in these models will now produce compilation errors. The only supported customization model is through extensions. If you cannot customize these models through extension, then you will have to make a request to Microsoft to enable extensibility by changing the standard application.

The following table includes a list of models that are now hard-sealed with this release.

<table>
<thead>
<tr>
<th>MODULE</th>
<th>MODEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeneralLedger</td>
<td>GeneralLedger</td>
</tr>
<tr>
<td>Ledger</td>
<td>Ledger</td>
</tr>
<tr>
<td>PersonnelManagement</td>
<td>PersonnelManagement</td>
</tr>
<tr>
<td>Retail</td>
<td>Retail</td>
</tr>
<tr>
<td>SourceDocumentation</td>
<td>SourceDocumentation</td>
</tr>
<tr>
<td>SourceDocumentationTypes</td>
<td>SourceDocumentationTypes</td>
</tr>
<tr>
<td>Subledger</td>
<td>Subledger</td>
</tr>
<tr>
<td>Tax</td>
<td>Tax</td>
</tr>
<tr>
<td>AccountsPayableMobile</td>
<td>AccountsPayableMobile</td>
</tr>
<tr>
<td>ApplicationWorkspaces</td>
<td>ApplicationWorkspaces</td>
</tr>
<tr>
<td>BankTypes</td>
<td>BankTypes</td>
</tr>
<tr>
<td>BusinessProcess</td>
<td>BusinessProcess</td>
</tr>
<tr>
<td>Calendar</td>
<td>Calendar</td>
</tr>
<tr>
<td>ContactPerson</td>
<td>ContactPerson</td>
</tr>
<tr>
<td>CostAccounting</td>
<td>CostAccounting</td>
</tr>
<tr>
<td>CostAccountingAX</td>
<td>CostAccountingAX</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Dimensions</td>
</tr>
<tr>
<td>DirectoryUpgrade</td>
<td>DirectoryUpgrade</td>
</tr>
<tr>
<td>DOM</td>
<td>DOM</td>
</tr>
<tr>
<td>ElectronicReporting</td>
<td>ElectronicReporting</td>
</tr>
<tr>
<td>MODULE</td>
<td>MODEL</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>ElectronicReportingAppSuiteIntegration</td>
<td>ElectronicReportingAppSuiteIntegration</td>
</tr>
<tr>
<td>ElectronicReportingCore</td>
<td>ElectronicReportingCore</td>
</tr>
<tr>
<td>ElectronicReportingDotNetUtils</td>
<td>ElectronicReportingDotNetUtils</td>
</tr>
<tr>
<td>ElectronicReportingForAx</td>
<td>ElectronicReportingForAx</td>
</tr>
<tr>
<td>ElectronicReportingMapping</td>
<td>ElectronicReportingMapping</td>
</tr>
<tr>
<td>ExpenseMobile</td>
<td>ExpenseMobile</td>
</tr>
<tr>
<td>FinancialReporting</td>
<td>FinancialReporting</td>
</tr>
<tr>
<td>FinancialReportingEntityStore</td>
<td>FinancialReportingEntityStore</td>
</tr>
<tr>
<td>FiscalBooks</td>
<td>FiscalBooks</td>
</tr>
<tr>
<td>InventoryDimensionConversion</td>
<td>InventoryDimensionConversion</td>
</tr>
<tr>
<td>Measurement</td>
<td>Measurement</td>
</tr>
<tr>
<td>PaymentPredictor</td>
<td>PaymentPredictor</td>
</tr>
<tr>
<td>PerformanceTool</td>
<td>PerformanceTool</td>
</tr>
<tr>
<td>Personnel</td>
<td>Personnel</td>
</tr>
<tr>
<td>PersonnelCore</td>
<td>PersonnelCore</td>
</tr>
<tr>
<td>PersonnelMobile</td>
<td>PersonnelMobile</td>
</tr>
<tr>
<td>PersonnelUpgrade</td>
<td>PersonnelUpgrade</td>
</tr>
<tr>
<td>Policy</td>
<td>Policy</td>
</tr>
<tr>
<td>Project</td>
<td>Project</td>
</tr>
<tr>
<td>ProjectMobile</td>
<td>ProjectMobile</td>
</tr>
<tr>
<td>RegulatoryServices</td>
<td>RegulatoryServices</td>
</tr>
<tr>
<td>SCMMobile</td>
<td>SCMMobile</td>
</tr>
<tr>
<td>SelfHealing</td>
<td>SelfHealing</td>
</tr>
<tr>
<td>SelfHealingRules</td>
<td>SelfHealingRules</td>
</tr>
<tr>
<td>SystemHealth</td>
<td>SystemHealth</td>
</tr>
</tbody>
</table>
## Enumerations that have been made extensible

The following changes were made to support extending enumerations:

- Many enumerations in the standard application have been made extensible. An enumeration is made extensible by setting two properties on the enumeration. The `IsExtensible` property is set to `Yes`, and the `UseEnumValue` property is set to `No`.
- Some enumerations represent state. New façade methods have been added to help enable adding enumeration values by extension. For information about how to extend an enumeration, see Add values to enums through extension.
- Some application code that uses enumerations was changed to support extensibility. Common changes include:
  - Removing `throw` exception statements in the default case of a switch to allow post-event subscription.
  - Adding `SysExtension` support for extension.
  - Adding explicit delegates.

### EnumERATION

<table>
<thead>
<tr>
<th>Enumerations</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssetCalendarYearType</td>
</tr>
<tr>
<td>AssetDepreciationConvention</td>
</tr>
<tr>
<td>AssetDepreciationMethod</td>
</tr>
<tr>
<td>AssetPeriodMonth</td>
</tr>
<tr>
<td>AssetSoldScrap</td>
</tr>
<tr>
<td>AssetStatusLVPFilter</td>
</tr>
<tr>
<td>AssetTransType</td>
</tr>
<tr>
<td>BaseDataProd</td>
</tr>
<tr>
<td>BOMRouteVersionSelect</td>
</tr>
<tr>
<td>BudgetPlanGenerateSource</td>
</tr>
<tr>
<td>BudgetPlanScenarioAccessLevel</td>
</tr>
<tr>
<td>BudgetPlanScenarioAttribute</td>
</tr>
<tr>
<td>BudgetTransactionColumnType</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>---------------------------------</td>
</tr>
<tr>
<td>BusinessEvent_ActivityJournal</td>
</tr>
<tr>
<td>BusinessEvent_CustomerInvoice</td>
</tr>
<tr>
<td>BusinessEventRelievingMethod</td>
</tr>
<tr>
<td>CollabSiteEntity_Type</td>
</tr>
<tr>
<td>CollabSiteSharePoint_Type</td>
</tr>
<tr>
<td>CostSheetNodeType</td>
</tr>
<tr>
<td>CreditCardAddEdit</td>
</tr>
<tr>
<td>CreditCardApprovalType</td>
</tr>
<tr>
<td>CreditCardDupCheckResult</td>
</tr>
<tr>
<td>CreditCardPaymType</td>
</tr>
<tr>
<td>CustAssessment</td>
</tr>
<tr>
<td>CustCollectionLetterCode</td>
</tr>
<tr>
<td>CustInterestFeeType</td>
</tr>
<tr>
<td>CustomerTransactionType</td>
</tr>
<tr>
<td>CustSettlementPriorityAttribute</td>
</tr>
<tr>
<td>CustSettlementTrans</td>
</tr>
<tr>
<td>CustTransRefType</td>
</tr>
<tr>
<td>CustVendDisputeStatus</td>
</tr>
<tr>
<td>CustVendForeignExchRefIndicator_US</td>
</tr>
<tr>
<td>CustVendGatewayOperatorOFACIndicator_US</td>
</tr>
<tr>
<td>CustVendorBlocked</td>
</tr>
<tr>
<td>CustVendOutPaymTrade</td>
</tr>
<tr>
<td>CustVendPaymStatus</td>
</tr>
<tr>
<td>CustVendSecondaryOFACIndicator_US</td>
</tr>
<tr>
<td>DimensionCacheScope</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>DirPartyRoleType</td>
</tr>
<tr>
<td>DirPartyRoleView</td>
</tr>
<tr>
<td>DistributionProcess</td>
</tr>
<tr>
<td>DistributionProcessingState</td>
</tr>
<tr>
<td>DistributionStatus</td>
</tr>
<tr>
<td>EProjUpdateSubProjStage</td>
</tr>
<tr>
<td>HcmPositionForecastStatusSelection</td>
</tr>
<tr>
<td>JournalizingDefinitionLedgerEntryTypeld</td>
</tr>
<tr>
<td>LedgerBalanceExportFieldSeparator</td>
</tr>
<tr>
<td>LedgerBalanceExportHeaders</td>
</tr>
<tr>
<td>LedgerBalanceExportHiddenFields</td>
</tr>
<tr>
<td>LedgerBalanceExportInvertSign</td>
</tr>
<tr>
<td>LedgerBalanceExportSubcomponents</td>
</tr>
<tr>
<td>LedgerBalanceExportSubtotals</td>
</tr>
<tr>
<td>LedgerColumnType</td>
</tr>
<tr>
<td>LedgerConsDim</td>
</tr>
<tr>
<td>LedgerConsolidateAccountSource</td>
</tr>
<tr>
<td>LedgerDataExportFormat</td>
</tr>
<tr>
<td>LedgerReconciliationStatus</td>
</tr>
<tr>
<td>LedgerShowCurrency</td>
</tr>
<tr>
<td>LedgerSIEFileType</td>
</tr>
<tr>
<td>LedgerTransactionType</td>
</tr>
<tr>
<td>LedgerTransEnigneBuildQuery</td>
</tr>
<tr>
<td>LogisticsAddressElement</td>
</tr>
<tr>
<td>LogisticsAddrZipCodeImportCountryRegion</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>LogisticsLocationEntityTypeName</td>
</tr>
<tr>
<td>LogisticsLocationSelectSourceType</td>
</tr>
<tr>
<td>MCROrderEventType</td>
</tr>
<tr>
<td>PaymManDocType</td>
</tr>
<tr>
<td>ProjAccountType</td>
</tr>
<tr>
<td>ProjAccountTypeCost</td>
</tr>
<tr>
<td>ProjAccountTypeSales</td>
</tr>
<tr>
<td>ProjActualVsForecastCategory</td>
</tr>
<tr>
<td>ProjActualVsForecastValue</td>
</tr>
<tr>
<td>ProjAlertType</td>
</tr>
<tr>
<td>ProjAssignConflictStatus</td>
</tr>
<tr>
<td>ProjCategoryType</td>
</tr>
<tr>
<td>ProjChoose</td>
</tr>
<tr>
<td>ProjContractType</td>
</tr>
<tr>
<td>ProjCostSales</td>
</tr>
<tr>
<td>ProjDimensionStrType</td>
</tr>
<tr>
<td>ProjectReportingAnalyticsWorkspaces</td>
</tr>
<tr>
<td>ProjEstimateMethod</td>
</tr>
<tr>
<td>ProjExportToExcelDimension</td>
</tr>
<tr>
<td>ProjFoundMethod</td>
</tr>
<tr>
<td>ProjFunctionType</td>
</tr>
<tr>
<td>ProjFundingRuleType</td>
</tr>
<tr>
<td>ProjInvoiceFrequency</td>
</tr>
<tr>
<td>ProjInvoiceProposalsTransSelectionTypes</td>
</tr>
<tr>
<td>ProjLevelFilterOption</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>ProjListLedgerTransType</td>
</tr>
<tr>
<td>ProjOrigin</td>
</tr>
<tr>
<td>ProjOriginOnAcc</td>
</tr>
<tr>
<td>projPostedProjectTransactionsListFilter</td>
</tr>
<tr>
<td>ProjProdTableListPageFilter</td>
</tr>
<tr>
<td>projProjectsListFilter</td>
</tr>
<tr>
<td>ProjQuotationTransTypeFilter</td>
</tr>
<tr>
<td>ProjResourceCapacityBooking</td>
</tr>
<tr>
<td>ProjResourceViewType</td>
</tr>
<tr>
<td>ProjSelectTransOnAcc</td>
</tr>
<tr>
<td>ProjServerProcessStatus</td>
</tr>
<tr>
<td>ProjStatementTypeSub</td>
</tr>
<tr>
<td>ProjStatus</td>
</tr>
<tr>
<td>ProjTransType</td>
</tr>
<tr>
<td>ProjValConnection</td>
</tr>
<tr>
<td>ProjValidateType</td>
</tr>
<tr>
<td>ProjViewSubProjects</td>
</tr>
<tr>
<td>ProjYearEndOptions</td>
</tr>
<tr>
<td>PSAActivityDisplayDefault</td>
</tr>
<tr>
<td>PSAActivityParent</td>
</tr>
<tr>
<td>PSAActivityParent</td>
</tr>
<tr>
<td>PSADetailLevel</td>
</tr>
<tr>
<td>PSAExpenseProjValCategoryType</td>
</tr>
<tr>
<td>PSAInvoiceFormats</td>
</tr>
<tr>
<td>PSAProjInvoiceDetailGrouping</td>
</tr>
<tr>
<td>PSAProjInvoiceDetailSortBy</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>PSAProjOriginakVsCurrent</td>
</tr>
<tr>
<td>PSAPWPAssessment</td>
</tr>
<tr>
<td>PSAResAssignView</td>
</tr>
<tr>
<td>PSAResSchedStatus</td>
</tr>
<tr>
<td>PurchStatus</td>
</tr>
<tr>
<td>QuotationProjTransType</td>
</tr>
<tr>
<td>ReasonCodeAccountType</td>
</tr>
<tr>
<td>ResBasicSearchType</td>
</tr>
<tr>
<td>ResRollUpResourceType</td>
</tr>
<tr>
<td>ResTransferType</td>
</tr>
<tr>
<td>ResUtilizationCategoryEnum</td>
</tr>
<tr>
<td>RetailEventNotificationType</td>
</tr>
<tr>
<td>SourceDocument_ActivityJournal</td>
</tr>
<tr>
<td>SourceDocumentLine_ActivityJournalLine</td>
</tr>
<tr>
<td>SpecType</td>
</tr>
<tr>
<td>SyncProjTableAddSubProj</td>
</tr>
<tr>
<td>SyncProjTableDeleteSubProj</td>
</tr>
<tr>
<td>TrvCarRentalChargeType</td>
</tr>
<tr>
<td>TrvExpenseFilter</td>
</tr>
<tr>
<td>TrvExpenseReportGroupBy</td>
</tr>
<tr>
<td>TrvIntermediatePageOnCreateExpenseReport</td>
</tr>
<tr>
<td>TrvPayrollQtyOrDays</td>
</tr>
<tr>
<td>TrvPBSTxtType</td>
</tr>
<tr>
<td>TrvPolicyViolationAction</td>
</tr>
<tr>
<td>TrvPosting</td>
</tr>
</tbody>
</table>
Foundation changes were made to improve support for extensible enumerations. The **SysPlugin** framework was enabled for enumerations where **IsExtensible** is set to **Yes**. Views were enabled with new name-based syntax for enumerations.

**Data manipulation methods that do not raise DataEvents or missing insert, update, delete pre- and post-data events**

As a general practice, you use data methods on tables to raise events that can be used for extending the application. The code base has not always followed this practice. For example, the **doInsert**, **doUpdate**, and **doDelete** data methods and certain table implementations did not make a call to **super()** in the data method.

The **insert**, **update**, and **delete** methods on the type classes have been refactored. Changes were made so that **super()** is called more consistently in data methods. These changes enable extensions to be added to these methods, so that pre- and post-events are now available for extension. The tables where the **insert**, **update**, and **delete** events were enabled for extension are listed in the following table.

---

### ENUMERATION

<table>
<thead>
<tr>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrvPostStatus</td>
</tr>
<tr>
<td>TrvTaxType</td>
</tr>
<tr>
<td>TrvUDFDisplayOption</td>
</tr>
<tr>
<td>TSAApprovalLevel</td>
</tr>
<tr>
<td>TSDocumentStatusReset</td>
</tr>
<tr>
<td>TSTimesheetFilter</td>
</tr>
<tr>
<td>TSTimesheetLineFilterType</td>
</tr>
<tr>
<td>TSTimesheetListPageFilters</td>
</tr>
<tr>
<td>TSVendorPerformanceThreshold</td>
</tr>
<tr>
<td>TypeOfCreditmaxCheck</td>
</tr>
<tr>
<td>VendInvoiceCloseCommand</td>
</tr>
<tr>
<td>VendorInvoiceSearchOptions</td>
</tr>
<tr>
<td>VendOutPaymFeeDistribution</td>
</tr>
</tbody>
</table>

---

### TABLE AND METHOD

<table>
<thead>
<tr>
<th>Table and Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOM</td>
</tr>
<tr>
<td>BOMTable</td>
</tr>
<tr>
<td>BOMVersion</td>
</tr>
</tbody>
</table>
TABLE AND METHOD

ProjTableType.delete

ProjTableType.update

Route

RouteOpr

RouteTable

RouteVersion

SalesLineType::interCompanyResetDeliverNow

Table ForecastSales

Exposing class members

Additional private members are now available for customization as a result of changes to access modifiers and parm methods. The chain of command platform feature enables extension class access to protected methods and members. For more information about chain of command, see Extensible X++: Chain of Command.

MEMBER

BankPaymCancel.custTransToCancel

CustCollectionLetterCancel - method queryBuildUpdate

CustCollectionLetterPost - method queryBuildUpdate

CustCollectionLetterPost - method updateFee

CustCollectionLetterPost - method validateCollectionLetter

CustInterestCancel - method updateQuery

CustInterestHelper - method getFeeLedgerAccount

CustInterestHelper - method getInterestRecord

CustInterestHelper - method getPostingProfile

CustInterestHelper - method getTransLedgerAccount

CustInterestHelper - method getTransLineLedgerAccount

CustInterestHelper - method getVerDetailLedgerDimensionByIntTrans

CustInterestPost - method postVoucher
CustOutPaymControlController

CustVendCreatePaymJournal - method dialogAddInvoiceSelectionCriteriaFields

CustVendPaymProposal - method createProposalLine

CustVendPaymProposal - method parmLedgerJournalId

CustVendPaymProposalLineInsertSetManager - variables

CustVendPaymProposalOrg - variables

CustVendPaymProposalTransferToJournal - method trackSpecTransForUpdate

CustVendPaymProposalTransferToJournal - variables

Form ProjWorkBreakdownStructure

Form/Class CustPaymModeSpec

Form/Class VendPaymModeSpec

InventUpd_ChildReference.initUpdate

InventUpd_ChildReference.parmInventDimId

LogisticsLocationFormHandler.callerGetAddressRecord

ProjAdjustmentSelect.newQuery.addAdditionalHeaderRange

ProjAdjustmentSelect.processProjCostTrans

ProjAdjustmentSplit.deleteTransaction

ProjAdjustmentSplit.splitTransaction

ProjInvoiceChoose.parmprojInvoiceProjId

ProjProposalTotals.projInvoiceExchRate

SalesInvoiceJournalCreateBase

smmActivityCreate.createOrPrompt

Table SalesQuotationTable.canSubmitToWorkflow

VendorInvoiceLineSourceDocLineItem.initializeProjectFields

WHSWorkUserSession.WorkExecuteMode
Construct methods with throw statements

Some construct methods were implemented with throw statements if there was a missing implementation for a given type. This doesn't work well with extensibility, so to mitigate this, construct methods were changed so that they do not throw exceptions. These methods are now to open for extensibility through class augmentation or by post-event subscription.

<table>
<thead>
<tr>
<th>OBJECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddressZipCodeImport</td>
</tr>
<tr>
<td>CaseCategoryHierarchyTree</td>
</tr>
<tr>
<td>CustInterestCancel</td>
</tr>
<tr>
<td>CustInterestHelper</td>
</tr>
<tr>
<td>CustInterestPost</td>
</tr>
<tr>
<td>CustOutPaymControlController</td>
</tr>
<tr>
<td>CustTransQueryBuild</td>
</tr>
<tr>
<td>CustVendCreatePaymJournal_Cust</td>
</tr>
<tr>
<td>CustVendFindSettlements</td>
</tr>
<tr>
<td>CustVendOpenTransBalances.new</td>
</tr>
<tr>
<td>CustVendOpenTransManager.initFromCaller</td>
</tr>
<tr>
<td>CustVendPaymProposalTransferToJournal</td>
</tr>
<tr>
<td>CustVendTransQueryBuild</td>
</tr>
<tr>
<td>Form PdsApprovedVendorList</td>
</tr>
<tr>
<td>Form WhsContainerTable.init</td>
</tr>
<tr>
<td>FormletterJournalCreate.newSalesJournalCreate</td>
</tr>
<tr>
<td>FormLetterJournalPost.newPostSales</td>
</tr>
<tr>
<td>InventUpd_Physical::newInventMovement</td>
</tr>
<tr>
<td>InventUpd_Physical::newProdReleaseLossProfit_RU</td>
</tr>
<tr>
<td>LogisticsLocationSelectForm</td>
</tr>
<tr>
<td>PdsApprovedVendorListCheck.newBasedOnTableType</td>
</tr>
<tr>
<td>ProjInvoiceChoose</td>
</tr>
</tbody>
</table>
## Find methods with throw statements

Some find methods were implemented with throw statements if there was a missing implementation for a given type. This does not work well with extensibility, so to mitigate this, find methods were changed so that they do not throw exceptions. These methods are now to open for extensibility through class augmentation or by post-event subscription.

### Extracted method to open for class extensions

The **Chain of Command** feature lets you create extension classes. Extensions classes offer a stronger way of extending than other options because they allow access to both protected and public methods and members. This provides more flexibility than extending through delegates or by pre or post event.

Within this group of changes, longer methods are extracted into smaller methods. The new methods have a more specific focus and you have more control over the scope of your extensions.

After the introduction of the **Chain of Command** feature, we suggest using extensibility by extracting methods instead of adding delegates because this approach provides a more versatile solution.

The following table lists the new methods that have been extracted and opened for building extension classes.

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssetPost</td>
</tr>
<tr>
<td>BankPrintTestCheque</td>
</tr>
<tr>
<td>CustCreditLimit.showErrorMsg</td>
</tr>
<tr>
<td>CustVendCheque</td>
</tr>
<tr>
<td>CustVendChequeSlipTextCalculator</td>
</tr>
<tr>
<td>Form CustBankAccounts</td>
</tr>
</tbody>
</table>

### Table of Objects

<table>
<thead>
<tr>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProjTrans</td>
</tr>
<tr>
<td>PurchReqAutoCreate.newAutoCreate</td>
</tr>
<tr>
<td>PurchTableForm_Project</td>
</tr>
<tr>
<td>SalesQuantity</td>
</tr>
<tr>
<td>SalesTotals</td>
</tr>
<tr>
<td>WHSReservation</td>
</tr>
</tbody>
</table>

### Table of Methods

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>TradePostalAddress.partyTable</td>
</tr>
<tr>
<td>METHOD</td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td>Form DirPartyQuickCreateForm.init</td>
</tr>
<tr>
<td>Form HierarchyDetail.contextChanged</td>
</tr>
<tr>
<td>Form HierarchyDetail: smmActivate: initValue</td>
</tr>
<tr>
<td>Form HierarchyNameLookoup: Hierarchy: init</td>
</tr>
<tr>
<td>Form LedgerJournalTransDimension.init</td>
</tr>
<tr>
<td>Form ProjInvoiceProposalDetail.editInvoiceFormat</td>
</tr>
<tr>
<td>Form SalesCopying.upDateRemainderCache</td>
</tr>
<tr>
<td>Form SalesQuotationProjLinkWizard-&gt; changeType</td>
</tr>
<tr>
<td>Form smmActivities: ResponsibleWorker_Overview: lookupReference</td>
</tr>
<tr>
<td>Form smmActivities: smmActivities::initValue</td>
</tr>
<tr>
<td>FormletterJournalPrint</td>
</tr>
<tr>
<td>HierarchyTemplateCopying.run</td>
</tr>
<tr>
<td>HierarchyTemplateCopying_CRM.copyActivity</td>
</tr>
<tr>
<td>HierarchyTemplateCopyingDialog.main</td>
</tr>
<tr>
<td>HierarchyTree</td>
</tr>
<tr>
<td>HierarchyTree.buildSubTree</td>
</tr>
<tr>
<td>InventDimCtrl_Frm_Mov_QualityOrder::mustEnableField</td>
</tr>
<tr>
<td>InventDistinctProductValidator.checkProductNotStopped</td>
</tr>
<tr>
<td>InventMovement.createProjLedgerForUpdateLedgerAdjust</td>
</tr>
<tr>
<td>InventTransferUpdShip::populateIssueReceiptDimensions</td>
</tr>
<tr>
<td>JournalFormTable</td>
</tr>
<tr>
<td>JournalizingDefinitionManager.newJournalizingDefinitionManagerCustomer</td>
</tr>
<tr>
<td>LedgerJournalCheckPost.checkJournal</td>
</tr>
<tr>
<td>LedgerJournalCheckPost.postJournal</td>
</tr>
<tr>
<td>LedgerJournalEngine.parmLedgerJournalTrans_Project</td>
</tr>
</tbody>
</table>
METHOD

LedgerJournalEngine_Server.addVoucher
LedgerJournalEngine_VendApprove.cancelVoucher
LedgerJournalizeReportDP_DE.processReport
LedgerJournalTransUpdateCust.checkAccountBlocked
LedgerJournalTransUpdateVend.checkVendorBlocked
LogisticsAddressFormatProcess.run
ProjAdjustmentUpdate.journalTableInsert
ProjAdjustmentUpdate_Post
projCategoryLookup.buildQuery_PSA_impl
ProjEstimate.add
ProjFormLetter_invoice.projPrintFormLetter
ProjIntercompanyVendorInvoiceCreator.createVendorInvoiceLine
ProjListTransDP.insertTmpProjTransList
ProjPostRevenueProposal.projTransCreate
projUnpostedTransactionsListPage.populateMenuFunction
PSAProjAndContractInvoiceController.preRunModifyContract
PSARetenetionRelease.run
PurchAutoCreate_SalesLine.setPurchTable
PurchCreateFromSalesOrder.run
PurchFormletterParmDataInvoice.createParmLinesAndTable
PurchTableForm_DlvScheduleSyncEnabled.syncDeliveryScheduleCommercialAttributes
ReqCalc.deleteItemRequirement
ReturnAcknowledgmentAndDocumentDP.insertIntoTempTable
ReturnAcknowledgmentAndDocumentDP.setReturnAckAndDocumentTemplate
SalesAgreementFormDatasourceManager.transferCustAccount
Changes using other methods to support extensibility

The group of changes in this section includes several different approaches to extensibility and represents the extensibility changes made before Chain of Command was introduced. Some of the approaches used are extracting methods, adding "stub" methods, adding delegates, changing access modifiers on methods, and using the SysExtension framework. Please consult the implementation in places required for your customization to determine if the approach taken will work for your customization. In future releases, this group will be small, because we will primarily be using Chain of Command.

METHOD

Table TSTimesheetTrans.updateFromTimesheetLineWeekUpdateTSTimesheetTrans

Table VendTable.lookupVendor

Table WHSWorkTable -> deleteAndCleanupWorkLines

Table WHSWorkTable -> SetBlankFields

Table WHSWorkTable -> SetFields

Table WrkCtrActivity.getCompanyContext

Tax.insertLineInInternal

TransactionReversal_Cust.fld900_1_modified

whsLoadTemplateAssignmentForm: WHSLoadTable::clicked

WhsWorkExecuteDisplay.getNextFormState

WHSWorkExecuteDisplay.setBatchDetails

WhsWorkExecuteDisplayReturnOrder.buildReturnOrder

WhsWorkExecuteDisplayReturnOrder.displayForm

METHOD

AccDistRuleSaleOfProductExtendedPrice.parmLedgerDimensionAllocList

AssetPost.createInventorySoldTransaction

AssetSplit.validate

AxSalesLine.doSave

AxSalesLine.setLineAmount

AxSalesQuotationLine.setLineAmount

BankPaymCancel.main
<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>BankPaymCancel.run</td>
</tr>
<tr>
<td>BankPaymCancel.serverRun</td>
</tr>
<tr>
<td>BomCalcJob.main</td>
</tr>
<tr>
<td>CustCollectionLetterCancel.main</td>
</tr>
<tr>
<td>CustCollectionLetterCancel.run</td>
</tr>
<tr>
<td>CustCollectionLetterCreate.checkCustTransOpen</td>
</tr>
<tr>
<td>CustCollectionLetterCreate.createJournal</td>
</tr>
<tr>
<td>CustCollectionLetterPost.updateFee</td>
</tr>
<tr>
<td>CustCollectionLetterPost.validate</td>
</tr>
<tr>
<td>CustCollectionsExcelStatement.setTransactionWorksheetRow</td>
</tr>
<tr>
<td>CustInterestCancel.run</td>
</tr>
<tr>
<td>CustInterestCreate</td>
</tr>
<tr>
<td>CustInterestCreate.dialog</td>
</tr>
<tr>
<td>CustInterestCreate.runOnce</td>
</tr>
<tr>
<td>CustInterestHelper.getFeeLedgerAccount</td>
</tr>
<tr>
<td>CustInterestHelper.getVerDetailLedgerDimensionByIntTrans</td>
</tr>
<tr>
<td>CustInterestPost.postVoucher</td>
</tr>
<tr>
<td>CustInterestPost.run</td>
</tr>
<tr>
<td>CustInterestPost.updateFee</td>
</tr>
<tr>
<td>CustInterestPost.validateInterestTrans</td>
</tr>
<tr>
<td>CustInvoiceSpecDP::insertIntoTempTable</td>
</tr>
<tr>
<td>CustOutPaymControlController.init</td>
</tr>
<tr>
<td>CustPostInvoiceJob.custPostInvoiceUpdate()</td>
</tr>
<tr>
<td>CustSettlementPriorityProcessing</td>
</tr>
<tr>
<td>CustSettlementPriorityProcessing.markAllSelected</td>
</tr>
</tbody>
</table>
**METHOD**

CustSettlementPriorityProcessing.markTransByCreditNoteOnBillingClasses

CustVendCreatePaymJournal.initBalances

CustVendOpenTransManager::updateOriginatorForMarkedTrans

CustVendPaymProposalCalcPaym.calcPaymDueDate

CustVendReversePosting.updateNow

CustVendSettle.mustOffsetOriginalSummaryDistributions

CustVendVoucher.initLedgerPosting

DimensionDerivationRule.buildDimensionCombination

DimensionDerivationRule.initialize

EcoResProductReleaseManager.release

EcoResProductReleaseSessionBatch.runJob

EcoResProductReleaseSessionManager.executeOnServer

EcoResProductVariantCreationMgr.buildVariantSuggestions()

Form BankPaymCancel.closeOK

Form BOMChangeLine.init

Form BOMConsistOf: BOMCreate

Form ConfigPartOf: EcoResConfiguration

Form CustBankAccounts.write

Form CustCollections.showAgingIndicator

Form CustOpenTrans.editMarkTrans

Form CustOpenTrans.updateDesignStatic

Form CustPaymEntry.editIsMarkedForSettlement

Form CustPaymEntry.write

Form CustSettlementPrioritySetup.active

Form CustTable.PrintManagement.clicked
<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form EcoResProductImage.init</td>
</tr>
<tr>
<td>Form EcoResProductImage.setProductRecId</td>
</tr>
<tr>
<td>Form HRMAbsenceRequest.init</td>
</tr>
<tr>
<td>Form LedgerJournalTransAccrual.enableFields</td>
</tr>
<tr>
<td>Form LedgerJournalTransAccrual.LedgerJournalTransAccrual.clicked</td>
</tr>
<tr>
<td>Form LedgerJournalTransCustPaym: LedgerJournalTrans.active</td>
</tr>
<tr>
<td>Form LedgerJournalTransDaily: SettlementButton.clicked</td>
</tr>
<tr>
<td>Form LedgerJournalTransDimension.init</td>
</tr>
<tr>
<td>Form MarkupTable.init</td>
</tr>
<tr>
<td>Form MCRItemListCopying.copyLines</td>
</tr>
<tr>
<td>Form PCProductModelVersion</td>
</tr>
<tr>
<td>Form ProjInvoiceProposalCreateLines.modifiedTransFilter</td>
</tr>
<tr>
<td>Form ProjTransItem: ProjItemTrans.salesAmount</td>
</tr>
<tr>
<td>Form PurchCreateFromSalesOrderSalesLine.included</td>
</tr>
<tr>
<td>Form ReqItemTable.init</td>
</tr>
<tr>
<td>Form SalesCopying.canClose</td>
</tr>
<tr>
<td>Form SalesCreateQuotation.setFieldsActive</td>
</tr>
<tr>
<td>Form SalesQuotationTable.init</td>
</tr>
<tr>
<td>Form SalesTable: SalesLine.write</td>
</tr>
<tr>
<td>Form SalesTable: SalesLine_DS.ItemId.lookup</td>
</tr>
<tr>
<td>Form VendEditInvoice</td>
</tr>
<tr>
<td>FormletterJournalPos::newPostSales</td>
</tr>
<tr>
<td>FormLetterJournalPost.post</td>
</tr>
<tr>
<td>FormLetterService.run</td>
</tr>
<tr>
<td>From ProjTable.init</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>HierarchyTemplateCopying.copyHierarchy</td>
</tr>
<tr>
<td>HrmAbsenceRequestAction.run</td>
</tr>
<tr>
<td>InterCompanyUpdateRemPhys_PurchLine::synchronizeExternal</td>
</tr>
<tr>
<td>InterCompanyUpdateRemPhys_PurchLine::synchronizeInternal</td>
</tr>
<tr>
<td>InterCompanyUpdateStatus_PurchLine::synchronizeExternal</td>
</tr>
<tr>
<td>InterCompanyUpdateStatus_PurchLine::synchronizeInternal</td>
</tr>
<tr>
<td>IntrastatTransfer::calcCustVendInvoiceTransQty</td>
</tr>
<tr>
<td>InventDim::dimReportStr</td>
</tr>
<tr>
<td>InventMov Transfer::checkUpdateEstimated</td>
</tr>
<tr>
<td>InventMov Transfer::defaultDimension</td>
</tr>
<tr>
<td>InventMovement::checkNotSubDelivery</td>
</tr>
<tr>
<td>InventQualityManagementCreate.createQualityOrder</td>
</tr>
<tr>
<td>InventTransferParmLine::createShipLines</td>
</tr>
<tr>
<td>InventTransferParmLine::initWithInventTransferParmTable</td>
</tr>
<tr>
<td>InventTransferUpdShip::updateInventTransferLine</td>
</tr>
<tr>
<td>InventUpd_Estimated::createEstimatedInventTrans</td>
</tr>
<tr>
<td>InventUpd_Financial::newInventTransferLineReceive</td>
</tr>
<tr>
<td>InventUpd_Financial::newInventTransferLineShip</td>
</tr>
<tr>
<td>InventUpd_Financial::updateFinancialIssue</td>
</tr>
<tr>
<td>InventUpd_Financial::updateFinancialReceipt</td>
</tr>
<tr>
<td>InventUpd_Phenomenological::newInventMovement</td>
</tr>
<tr>
<td>InventUpd_Reservation::updateReserveBuffer</td>
</tr>
<tr>
<td>InventUpd_Reservation::updateReserveFromForm</td>
</tr>
<tr>
<td>InventUpdate::whsUpdateDimReservePhysical</td>
</tr>
<tr>
<td>InventUpdate::whsUpdateWorkTransDimIssue</td>
</tr>
<tr>
<td>METHOD</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>LedgerJournalCheckPost.postJournal</td>
</tr>
<tr>
<td>LedgerJournalEngine - method preDelete</td>
</tr>
<tr>
<td>LedgerJournalEngine::findSettledAmount</td>
</tr>
<tr>
<td>LedgerJournalEngine_CustPayment.allowEditTrans</td>
</tr>
<tr>
<td>LedgerJournalEngine_CustPayment.initDefaultDimension</td>
</tr>
<tr>
<td>LedgerJournalEngine_CustPayment.write</td>
</tr>
<tr>
<td>LedgerJournalFormTable.verifyCanDelete</td>
</tr>
<tr>
<td>LedgerVoucherTransObject.newTransLedgerJournal</td>
</tr>
<tr>
<td>LogisticsPostalAddressFormHandler.main</td>
</tr>
<tr>
<td>Map SalesPurchLine.calcPrice2LineAmount</td>
</tr>
<tr>
<td>Map SalesPurchLine.setPriceAgreement</td>
</tr>
<tr>
<td>Markup.calc</td>
</tr>
<tr>
<td>MarkupAdjustment::main</td>
</tr>
<tr>
<td>McrPriceHistoryForm.insertPotentialTradeAgreements</td>
</tr>
<tr>
<td>OffsetVoucherCust.updateNow</td>
</tr>
<tr>
<td>PcGenerateBOMTableAndVersion.generate</td>
</tr>
<tr>
<td>PriceDisc.findDisc</td>
</tr>
<tr>
<td>PriceDisc::findItemLineDiscAgreement</td>
</tr>
<tr>
<td>PriceDisc::findItemPriceAgreement</td>
</tr>
<tr>
<td>PriceDisc::findMultiLineDiscServer</td>
</tr>
<tr>
<td>PriceDisc::newFromPriceDiscHeading</td>
</tr>
<tr>
<td>PriceDisc.LineDisc::findLineDiscAgreement</td>
</tr>
<tr>
<td>PriceDisc_Price::findPriceAgreement</td>
</tr>
<tr>
<td>PriceDiscAdmCheckPost.checkJournal</td>
</tr>
<tr>
<td>PrintMgmtHierarchy_Project.getParentImplementation</td>
</tr>
<tr>
<td>METHOD</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>ProjAdjustmentSplit.createNewTrans.getNewTotalCostAmount</td>
</tr>
<tr>
<td>ProjInvoiceJournalPost.createProjInvoiceRevenue</td>
</tr>
<tr>
<td>ProjInvoiceProposalInsertLines.doSalesLine</td>
</tr>
<tr>
<td>ProjPost.postCost</td>
</tr>
<tr>
<td>ProjPostCostJournal.projTransCreate</td>
</tr>
<tr>
<td>ProjPostCostTrans_AdjNeg.projTransCreate</td>
</tr>
<tr>
<td>PurchAutoCreate_PurchReq.prepareSort</td>
</tr>
<tr>
<td>PurchCalcItem.initListBOM</td>
</tr>
<tr>
<td>PurchFormLetter.mainOnServer</td>
</tr>
<tr>
<td>PurchFormLetterParmData::newChooseLines</td>
</tr>
<tr>
<td>PurchFormletterParmDataInvoice.createParmLineAndSubLines</td>
</tr>
<tr>
<td>PurchInvoiceJournalPost.checkSourceLine</td>
</tr>
<tr>
<td>PurchInvoiceJournalPost.postCustVend</td>
</tr>
<tr>
<td>PurchInvoiceJournalPost.postInventory</td>
</tr>
<tr>
<td>PurchLineType.initDimensionsSpecificDefaulting</td>
</tr>
<tr>
<td>PurchLineType.interCompanyMirror</td>
</tr>
<tr>
<td>ReqCalcExplodeSales.run</td>
</tr>
<tr>
<td>SalesAutoCreate_ReleaseOrder.createSalesLine</td>
</tr>
<tr>
<td>SalesConfirmDP::createData</td>
</tr>
<tr>
<td>SalesConfirmDP::printDimHistory</td>
</tr>
<tr>
<td>SalesConfirmDP::setSalesConfirmDetailsTmp</td>
</tr>
<tr>
<td>SalesCopying.copy</td>
</tr>
<tr>
<td>SalesFormLetter.mainOnServer</td>
</tr>
<tr>
<td>SalesFormletterParmDataPickingList.insertParmLine</td>
</tr>
<tr>
<td>METHOD</td>
</tr>
<tr>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>SalesInvoiceController::main</td>
</tr>
<tr>
<td>SalesInvoiceDP::insertIntoSalesInvoiceTmp</td>
</tr>
<tr>
<td>SalesInvoiceDP::insertIntoSalesInvoiceTmp</td>
</tr>
<tr>
<td>SalesInvoiceDP::loadCustPackingSlipTrans</td>
</tr>
<tr>
<td>SalesInvoiceDP::createData</td>
</tr>
<tr>
<td>SalesInvoiceJournalCreate::initInvoiceLineFromSourceLine</td>
</tr>
<tr>
<td>SalesInvoiceJournalPost::postCustVend</td>
</tr>
<tr>
<td>SalesLine::initReleasedProductSpecificDefaulting</td>
</tr>
<tr>
<td>SalesLineType::initDimensionsSpecificDefaulting</td>
</tr>
<tr>
<td>SalesLineType::interCompanyMirror</td>
</tr>
<tr>
<td>SalesLineType::checkDelete</td>
</tr>
<tr>
<td>SalesLineType::delete</td>
</tr>
<tr>
<td>SalesLineType::insert</td>
</tr>
<tr>
<td>SalesLineType::interCompanyMirror</td>
</tr>
<tr>
<td>SalesLineType::setSalesStatus</td>
</tr>
<tr>
<td>SalesLineType::update</td>
</tr>
<tr>
<td>SalesLineType::validateWrite</td>
</tr>
<tr>
<td>SalesPickingListJournalCreate::createJournalLine</td>
</tr>
<tr>
<td>SalesQuotationConfirmationDP::processReport</td>
</tr>
<tr>
<td>SalesQuotationCopying.copy</td>
</tr>
<tr>
<td>SalesQuotationDP::processReport</td>
</tr>
<tr>
<td>SalesQuotationLine.modifySalesQty</td>
</tr>
<tr>
<td>SalesQuotationLineType.initReleasedProductSpecificDefaulting</td>
</tr>
<tr>
<td>SalesQuotationToLineField.getFieldDescription</td>
</tr>
<tr>
<td>SalesTable2LineUpdate.update</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>SalesTableListPageInteraction.setButtonInterCompany</td>
</tr>
<tr>
<td>SalesTableListPageInteraction.setButtonInvoice</td>
</tr>
<tr>
<td>SalesTableListPageInteraction.setButtonPickAndPack</td>
</tr>
<tr>
<td>SalesUpdateRemain</td>
</tr>
<tr>
<td>SalesUpdateRemain::updateDeliveryRemain</td>
</tr>
<tr>
<td>smmActivityCreate.setup</td>
</tr>
<tr>
<td>smmAttendeeTable.insert</td>
</tr>
<tr>
<td>smmSalesCustItemStatisticsDP::processReport</td>
</tr>
<tr>
<td>SpecTransManager.updateFullSettlement</td>
</tr>
<tr>
<td>SubledgerJournalizer.loadAccDistTmpRelieveAccrual</td>
</tr>
<tr>
<td>SubledgerJournalizer.loadaccountingDistributionTmp</td>
</tr>
<tr>
<td>SubledgerJournalizer.recordSubledgerJourAccEntriesForRounding</td>
</tr>
<tr>
<td>SubledgerJournalizer.recordSubledgerJournalAccountEntries</td>
</tr>
<tr>
<td>SubLedgerJournalTransferUIBuilder::build</td>
</tr>
<tr>
<td>SuppItemCreate_SalesQuotation::createLine</td>
</tr>
<tr>
<td>Table - PurchLine.Insert</td>
</tr>
<tr>
<td>Table - PurchLine.Update</td>
</tr>
<tr>
<td>Table CostingVersion.validateField</td>
</tr>
<tr>
<td>Table CustBankAccount.lookupBankAccount</td>
</tr>
<tr>
<td>Table CustCollectionLetterJour.cancelCollectionLetterCodeCustTrans</td>
</tr>
<tr>
<td>Table CustInterestJour.feeLedgerDimension</td>
</tr>
<tr>
<td>Table CustInvoiceTable - method validateWrite</td>
</tr>
<tr>
<td>Table CustTable.blocked</td>
</tr>
<tr>
<td>Table CustTrans.reverseTransact</td>
</tr>
<tr>
<td>Table InventNonConformanceTable.setEditableFields</td>
</tr>
<tr>
<td>METHOD</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Table InventPosting.accountItemLedgerDimension</td>
</tr>
<tr>
<td>Table InventTable.updateAutoSalesPrice</td>
</tr>
<tr>
<td>Table InventTestAssociationTable.checkDocumentType</td>
</tr>
<tr>
<td>Table InventTrans.accountLossProfitLedgerDimension</td>
</tr>
<tr>
<td>Table LedgerJournalTrans.checkVATNumJournal</td>
</tr>
<tr>
<td>Table MarkupTrans.checkMarkCode</td>
</tr>
<tr>
<td>Table PurchLine.convertCurrencyCode</td>
</tr>
<tr>
<td>Table ReqPO.validateWrite</td>
</tr>
<tr>
<td>Table SalesLine.checkAndUpdateLoadLines</td>
</tr>
<tr>
<td>Table SalesLine.setPriceDisc</td>
</tr>
<tr>
<td>Table SalesQuotationLine.setPriceDisc</td>
</tr>
<tr>
<td>Table SalesTable.createMarkupTrans</td>
</tr>
<tr>
<td>Table TmpInventTransMark.updateTmpMark</td>
</tr>
<tr>
<td>Table TMSAppointment.validateWrite</td>
</tr>
<tr>
<td>Table WHSLoadLine::inventTransferLine</td>
</tr>
<tr>
<td>Table WHSLoadLine::purchLine</td>
</tr>
<tr>
<td>Table WHSLoadLine::salesLine</td>
</tr>
<tr>
<td>Table WHSRFMenuitemTable.validateWrite</td>
</tr>
<tr>
<td>Tax.distributeTotalTax</td>
</tr>
<tr>
<td>TradeInterCompany::insertInterCompanyInventDim</td>
</tr>
<tr>
<td>TransactionReversal_Asset.checkStatusApplicable</td>
</tr>
<tr>
<td>TransactionReversal_Cust.main</td>
</tr>
<tr>
<td>TransactionReversal_Cust.reversal</td>
</tr>
<tr>
<td>TransactionReversal_CustVend.createCustVendTrans</td>
</tr>
<tr>
<td>TSTimesheetLineWeek.loadFromLine</td>
</tr>
</tbody>
</table>
**Methods made hookable**

Extensibility support has been extended for some methods that were not public and were not hookable. The following methods have been explicitly decorated with hookable behavior.

<table>
<thead>
<tr>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>VendInvoiceInfoListPageMultiSelect.determineSelectState</td>
</tr>
<tr>
<td>WHSInventReserveDeltaLevelsEnumerator::moveNext</td>
</tr>
<tr>
<td>WhsPostEngineBase::createLoadFromShipment</td>
</tr>
<tr>
<td>WHSWorkCreateProdPut.insertProdParmforProdItem</td>
</tr>
<tr>
<td>WmsArrivalCreateJournal.createWMSJournalTransFromTmp</td>
</tr>
<tr>
<td>WMSPickingList_OrderPick.RunPrintMgmt</td>
</tr>
<tr>
<td>WrkCtrlScheduler_Prod.loadJobsDetail</td>
</tr>
<tr>
<td>Bank.checkBankIBAN</td>
</tr>
<tr>
<td>BankDepositCreateCancelJour.initValues</td>
</tr>
<tr>
<td>BankDepositCreateCancelJour.newDepositCreateCancelJour</td>
</tr>
<tr>
<td>BankPaymCancel.initParms</td>
</tr>
<tr>
<td>BankPaymCancel.updateCollectionsStatusAutomation</td>
</tr>
<tr>
<td>CustAccountStatementExtController</td>
</tr>
<tr>
<td>CustAccountStatementIntDP.insertCustAccountStatementIntTmp()</td>
</tr>
<tr>
<td>CustCollectionLetterPost.updateFee</td>
</tr>
<tr>
<td>CustInterestCreate.construct</td>
</tr>
<tr>
<td>CustProvisionalBalanceDP.insertCustProvisionalBalanceTmp()</td>
</tr>
<tr>
<td>CustSettlementPriorityProcessing</td>
</tr>
<tr>
<td>CustVendCreatePaymJournal.dialogAddDateSelectionFields</td>
</tr>
<tr>
<td>CustVendPaymReconciliationSetStatus</td>
</tr>
<tr>
<td>CustVendReversePosting.updateCustVendTrans</td>
</tr>
<tr>
<td>DataEntity EcoResTrackingDimensionGroupEntity.dataSourceDimensionFieldId</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>EcoResProductCrossTableManagers.saveValuesToProduct</td>
</tr>
<tr>
<td>EcoResProductImage.getImageFrom2Records</td>
</tr>
<tr>
<td>EUSalesListTransfer - 3 methods</td>
</tr>
<tr>
<td>Form EcoResProductCreate.applyTemplate</td>
</tr>
<tr>
<td>Form EcoResProductCreate.createData2Controls</td>
</tr>
<tr>
<td>Form PriceDiscTable.initFromCallerTable</td>
</tr>
<tr>
<td>Form ProjCostControl.setButtonVisibility</td>
</tr>
<tr>
<td>Form projPostedTransRellInfoFormPart: ProjPostTransView: costPrice</td>
</tr>
<tr>
<td>Form ProjTable.lookup Reference</td>
</tr>
<tr>
<td>Form ProjWorkBreakdownStructure</td>
</tr>
<tr>
<td>Form WHSPack.updateSummaryFields</td>
</tr>
<tr>
<td>FormletterJournalPost</td>
</tr>
<tr>
<td>FreeTextInvoiceDP.bankGroupIdName_CH</td>
</tr>
<tr>
<td>FreeTextInvoiceDP.bankZipCode_CH</td>
</tr>
<tr>
<td>FreeTextInvoiceDP.insertGiroInformation</td>
</tr>
<tr>
<td>FreeTextInvoiceDP.insertIntoFreeTextInvoiceHeaderFooterTmp</td>
</tr>
<tr>
<td>FreeTextInvoiceDP.insertIntoFreeTextInvoiceLocalizationTmp</td>
</tr>
<tr>
<td>FreeTextInvoiceDP.insertIntoFreeTextInvoiceTmp</td>
</tr>
<tr>
<td>HierarchyCreate_CRM.initHierarchy</td>
</tr>
<tr>
<td>HierarchyTemplateCopying_Proj.copyEstimates</td>
</tr>
<tr>
<td>InventDimGroupSetup.combineInventDimParms</td>
</tr>
<tr>
<td>InventLookupItemIdByDefaultOrder.initializeQuery</td>
</tr>
<tr>
<td>InventStorageDimMap.modifiedInventSiteFromParent</td>
</tr>
<tr>
<td>InventUpd_Physical.updatePhysicalReceiptTrans</td>
</tr>
<tr>
<td>JournalFormTable.initJournalTypeFromCaller</td>
</tr>
</tbody>
</table>
METHOD

LedgerJournalCheckPost.runInternal

Map SalesPurchLine.setPriceAgreement

Maps VendDocumentLineMap.setPurchaseInventReceiveNow

OffsetVoucherCust.getAutoSettlementQuery

ProjAdjustmentSelect.doTransCost

ProjAdjustmentSelect.doTransSale

ProjAdjustmentSelect.processProjEmpTrans

ProjAdjustmentSelect.validate

ProjAdjustmentSplit

ProjAdjustmentSplit.createNewTrans

ProjAdjustmentSplit.run

ProjAdjustmentUpdate.newPostAdjustment

ProjBegBalJournalTrans_CostSales.createProjTransPosting

ProjBegBalJournalTrans_Fee.createProjTransPosting

ProjBegBalJournalTrans_OnAcc.createProjTransPosting

projCostControl.progressUpdate

ProjEstimatesDataContract.setRevenueSalesPrice

ProjForecastBudget.forecastCopy

ProjForecastBudget.forecastDelete

ProjForecastPostItemFixedInvest.checkEnterCost

ProjForecastTransferFromWBS.transferToForecast

ProjFormLetter.mainOnServer

ProjFormLetter.printPreview

ProjInvoiceDP.insertIntoProjInvoiceLocalizationTmp

ProjInvoiceDP.insertIntoProjInvoiceTmp
<table>
<thead>
<tr>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProjInvoiceJournalCreate.creditMaxOk</td>
</tr>
<tr>
<td>ProjInvoiceJournalPost.initProposalUpdate</td>
</tr>
<tr>
<td>ProjLedger.newInventCost</td>
</tr>
<tr>
<td>ProjPlanVersionManager.copyActivityData</td>
</tr>
<tr>
<td>ProjPlanVersionsManager.createDraftVersion</td>
</tr>
<tr>
<td>ProjProjectTransListPageInteraction.linkActive</td>
</tr>
<tr>
<td>ProjTask.getCorrespondingTaskElementNumber</td>
</tr>
<tr>
<td>ProjValCheckTrans.validateMandatory</td>
</tr>
<tr>
<td>projWbsUpdateController.getNodesMapSortedByPath</td>
</tr>
<tr>
<td>PSAProjInvoiceDP.processLinesFromInvoiceJournal</td>
</tr>
<tr>
<td>PSAProjQuotationSubmitSend.validateProjectDates</td>
</tr>
<tr>
<td>PSAQuotationsDP.insertPSAQuotationsTmp</td>
</tr>
<tr>
<td>PurchaseOrderResponseCreate.createPurchaseOrderResponseLines</td>
</tr>
<tr>
<td>PurchCancel.cancelMarkup</td>
</tr>
<tr>
<td>PurchCreateFromSalesOrder.preMatchIncludedLinesWithAgreements</td>
</tr>
<tr>
<td>PurchPackingSlipDP.setPurchPackingSlipDetailsTmp</td>
</tr>
<tr>
<td>PurchPackingSlipDP.setPurchPackingSlipHeaderTmp</td>
</tr>
<tr>
<td>PurchReceiptsListDP.setPurchReceiptsListDetailsTmp</td>
</tr>
<tr>
<td>PurchReceiptsListDP.setPurchReceiptsListHeaderTmp</td>
</tr>
<tr>
<td>PurchSummary.checkFormLetterId</td>
</tr>
<tr>
<td>ReqPlanCopy.insertLog</td>
</tr>
<tr>
<td>ResRollupActivityWriter::updateRollupTableWithLockedCapacityForActivityResource()</td>
</tr>
<tr>
<td>ResRollupAvailabilityWriter.updateRollupTableWithLockedCapacityForNamedResource()</td>
</tr>
<tr>
<td>SalesConfirmDP.setSalesConfirmDetailsTmp</td>
</tr>
<tr>
<td>SalesConfirmDP.setSalesConfirmHeaderTmp</td>
</tr>
<tr>
<td>METHOD</td>
</tr>
<tr>
<td>-------------------------</td>
</tr>
<tr>
<td>SalesInvoiceController::main</td>
</tr>
<tr>
<td>SalesInvoiceDP.bankGroupIdName_CH</td>
</tr>
<tr>
<td>SalesInvoiceDP.bankZipCode_CH</td>
</tr>
<tr>
<td>SalesInvoiceDP.insertIntoSalesInvoiceHeaderFooterTmp</td>
</tr>
<tr>
<td>SalesInvoiceDP.insertIntoSalesInvoiceLocalizationTmp</td>
</tr>
<tr>
<td>SalesInvoiceDP.insertIntoSalesInvoiceTmp</td>
</tr>
<tr>
<td>SalesPackingSlipDP.setSalesPackingSlipDetailsTmp</td>
</tr>
<tr>
<td>SalesPackingSlipDP.setSalesPackingSlipHeaderTmp</td>
</tr>
<tr>
<td>SalesQuotationLineType.validateDelete</td>
</tr>
<tr>
<td>SalesQuotationLineType.validateWrite</td>
</tr>
<tr>
<td>SalesQuotationTableForm.CreateABSFromTemplate</td>
</tr>
<tr>
<td>salesQuotationTransferToProject.initParameters</td>
</tr>
<tr>
<td>SalesTable.initFromCustTableMandatoryFields</td>
</tr>
<tr>
<td>SalesTableListPageInteraction.setButtonSell</td>
</tr>
<tr>
<td>smmActivityCreate.createActivity</td>
</tr>
<tr>
<td>smmActivityCreate.new</td>
</tr>
<tr>
<td>smmActivityParentLinkTablee.insert</td>
</tr>
<tr>
<td>SubledgerJournalizerProjectExtension.createLedgerUpdate</td>
</tr>
<tr>
<td>Table CustBankAccount.validatePreNote</td>
</tr>
<tr>
<td>Table InventItemGTIN.formatGTIN</td>
</tr>
<tr>
<td>Table PriceDiscAdmTrans.checkItemRelation</td>
</tr>
<tr>
<td>Table PriceDiscAdmTrans.checkProductDimensions</td>
</tr>
<tr>
<td>Table ProjCategory.lookupProjCategoryType</td>
</tr>
<tr>
<td>Table ProjTable.validateWriteServer</td>
</tr>
<tr>
<td>Table PSAActivityEstimates.checkupdateQuotationLine</td>
</tr>
</tbody>
</table>
Table PSAActivityEstimates.setSalesPriceFromCostPrice
Table PurchLine.setPriceDisc
Table PurchTable.internalTableIdToTableId_W
Table SalesLine.setPriceAgreement
Table Salesline.setPriceDisc
Table SalesQuotationLine.setPriceAgreement
Table SalesQuotationLine.setPriceDisc
Table SalesTable.setSalesOrderReleaseStatus
Table TmpCustVendTrans.createLineCreditLimit
Table TmpCustVendTrans.createLineCreditRemain
Table TmpCustVendTrans.createLineOrdered
Table TmpCustVendTrans.createLinePackingSlip
Table TmpCustVendTrans.createLineTotal
Table TmpCustVendTrans.insertTmpCustVendTransForCustBalance
Table TSTimeSheetLine.checkActivity
TSTimesheetLine::buildQuerySmmActivities
TsTimesheetPost.validatePost
VendProvisionalBalanceDP.insertVendProvisionalBalanceTmp()
VendTransQueryBuild::construct
VersioningPurchaseOrderResponse.archiveResponseLines
VersioningPurchaseOrderResponse.restoreLines
WhsCycleCountCreateLocation.run
WhsLoadReplenishment.calculateReplenishQty
WHSLoadTable::initPurchOriginDestination
WhsReplenishment.calculateReplenishQty
Inline delegates

Inline delegates are now available. The most common way to use inline delegates is to split the method into more granular methods and enable extensibility events in the smaller methods.
METHOD

BOMCalcItem.initListBOM

BOMCalcJob.runBOMCalculation

BOMRouteCopyJob.checkTo

BOMRouteCopyJob.main

BomRouteCopyJob::main

BomSearch.init

bomVersionActivate.run

CaseDetailForm.lookupParentCase

CaseDetailFormCreate.main

CaseUpdateStatus.changeStatus

CaseUpdateStatus_Close.updateStatus

ChequeDP.insertChequeTmp

Class SalesLineType.intercompanyMirror

Commission.run

CostControlPostingSourceDocumentLine.createCommittedCost

CostSheetPanel.build

CustAccountStatementExtController.insertCustAccountStatementExtTmp

CustAgingReportController.getReportName

CustAgingReportDP.insertCustAgingReportTmp

CustCollectionJourDP.collectionLetterTitle

CustCollectionJourDP.insertCustCollectionJourTmp

CustCollectionLetterCancel.main

CustCollectionLetterCreate.createJournal

CustCollectionLetterCreate.updateCreatedCollectionLetter

CustCollectionLetterPost.run
<table>
<thead>
<tr>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustCollectionLetterPost.updateFee</td>
</tr>
<tr>
<td>CustInterestCancel.run</td>
</tr>
<tr>
<td>CustInterestCreate.createJournal</td>
</tr>
<tr>
<td>CustInterestCreate.insertCustInterestTrans</td>
</tr>
<tr>
<td>CustInterestCreate.insertCustInterestTransLine</td>
</tr>
<tr>
<td>CustInterestPost.updateCustInterestTransVoucherRef</td>
</tr>
<tr>
<td>CustInterestPost.updateFee</td>
</tr>
<tr>
<td>CustInvoiceDP::insertCustInvoiceTmp</td>
</tr>
<tr>
<td>CustInvoiceSpecDP::insertIntoTempTable</td>
</tr>
<tr>
<td>CustNsf.createFeeJournalTrans</td>
</tr>
<tr>
<td>CustOutPaymControlController.insert</td>
</tr>
<tr>
<td>CustPostInvoice.createJournalHeader</td>
</tr>
<tr>
<td>CustPostInvoice::createJournalHeader</td>
</tr>
<tr>
<td>CustSettlementPriorityProcessing.createTempData</td>
</tr>
<tr>
<td>CustSettlementPriorityProcessing.markTransByCreditNoteOnBillingClasses</td>
</tr>
<tr>
<td>CustTransOpenPerDateDP::insertCustTransOpenPerDateTmp</td>
</tr>
<tr>
<td>CustVendCreatePaymJournal.checkBlocked</td>
</tr>
<tr>
<td>CustVendCreatePaymJournal.dialogAddInvoiceSelectionCriteriaFields</td>
</tr>
<tr>
<td>CustVendCreatePaymJournal.runPaymentProposalGenerationProcess</td>
</tr>
<tr>
<td>CustVendCreatePaymJournal_Vend.UpdateQuery</td>
</tr>
<tr>
<td>CustVendFindSettlements.findSettledSettlements</td>
</tr>
<tr>
<td>CustVendOpenTransBalances.initAccountNumCurrencies</td>
</tr>
<tr>
<td>CustVendOpenTransBalances.new</td>
</tr>
<tr>
<td>CustVendOpenTransManager.initFromCaller</td>
</tr>
<tr>
<td>CustVendOpenTransManager.updateOriginatorForMarkedTrans</td>
</tr>
<tr>
<td>METHOD</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>CustVendPaymProposal.createProposalLine</td>
</tr>
<tr>
<td>CustVendPaymProposalTransferToJournal.initLedgerJournalTransFromPaymLine</td>
</tr>
<tr>
<td>CustVendPaymProposalTransferToJournal.run</td>
</tr>
<tr>
<td>CustVendPaymProposalTransferToJournal.transferProposal</td>
</tr>
<tr>
<td>CustVendReversePosting.updateCustVendTrans</td>
</tr>
<tr>
<td>CustVendSettle.createSettlementForDebitOrCreditTrans</td>
</tr>
<tr>
<td>CustVendSumForPaym::Validate</td>
</tr>
<tr>
<td>CustVendVoucher.post</td>
</tr>
<tr>
<td>CustVoucher.createInvoiceJournal</td>
</tr>
<tr>
<td>DataEntityView FreeTextInvoiceEntity.insertFreeTextInvoiceLines</td>
</tr>
<tr>
<td>DataEntityView FreeTextInvoiceEntity.preTargetProcessSetBased</td>
</tr>
<tr>
<td>DimensionHierarchyHelper::getHierarchyTypeByAccountType</td>
</tr>
<tr>
<td>EcoResProductMasterManager.addProductDimensionValue</td>
</tr>
<tr>
<td>EcoResProductReleaseManager.createInventTable</td>
</tr>
<tr>
<td>EcoResProductReleaseManager.createInventItemSetupSupplyType</td>
</tr>
<tr>
<td>EcoResProductReleaseManager.setInventTableFields</td>
</tr>
<tr>
<td>EcoResProductTemplateManager.getBufferByDataSourceName</td>
</tr>
<tr>
<td>EcoResProductVariantManager.createProductVariant</td>
</tr>
<tr>
<td>Extend delegatestr(DirPartyPostalAddressFormHandler, defaultLocationRoles_delegate)</td>
</tr>
<tr>
<td>Form BankReconciliation: BankAccountReconcile::clicked</td>
</tr>
<tr>
<td>Form CustCreditLimitCreditPart.totalAgingByCompany</td>
</tr>
<tr>
<td>Form CustDirectDebitMandate.run</td>
</tr>
<tr>
<td>Form CustFormletterParameters.PrintMgMt.clicked</td>
</tr>
<tr>
<td>Form CustOpenTrans.init</td>
</tr>
<tr>
<td>Form CustOpenTrans.updateDesignStatic</td>
</tr>
</tbody>
</table>
METHOD

Form CustOpenTrans: Button UpdateNow::clicked

Form CustOpenTrans::doesCallerAllowEdit

Form CustTable: CustTable::write

Form EcoResProductCreate.writeMoreFields

Form EcoResProductVariantsPerCompany: InventDimCombination::write

Form HierarchyTemplateCopying_Prop.copyEstimates

Form InventDimParmFixed: InventDimParm::create

Form InventOnhandReserve: InventSum::reserveNow

Form InventOnhandReserve: InventTransOriginMovement::movementOnOrderUnit

Form InventOnhandReserve: InventTransOriginMovement::movementReservOrderedUnit

Form InventOnhandReserve: InventTransOriginMovement::movementReservPhysicalUnit

Form InventTransRegister: TmpInventTransWMS::setEnabled

Form LedgerJournalTransCustPaym.accountNumModifiedPost

Form LedgerJournalTransCustPaym: Button ButtonSettlement::clicked

Form LedgerJournalTransVendPaym: buttonPaymReconciliation::Clicked

Form LedgerJournalTransVendPaym: PaymReconciliationReject::Clicked

Form MarkupTrans.MarkupTrans_DS.active()

Form MCRSalesQuickQuote.init

Form MCRSalesQuickQuote.prepareSearch

Form MCRSalesQuickQuote.tmpFrmVirtualInventDimId

Form MRCSalesQuickQuote.createLines

Form PriceDiscActual::init

Form ProcCategoryHierarchyManagement.init

Form ProjAdjustment.init

Form ProjAdjustment.selectAdjRecords
Form ProjCreditNoteSelect.canClose
Form ProjCreditNoteSelect.writeTmpFrmVirtual
Form ProjInvoiceProposalCreateLines.TransTypeSelectionCtrl.lookup
Form PurchTable: PurchTable::enableJournalButtons
Form SalesATPSalesATP
Form SalesQuickQuote: InventDimCombination::setQuantyties
Form SalesQuickQuote: InventDimCombination::salesQty
Form SalesQuotationProjTable::SalesQuotationLine::ItemId::modified
Form SalesQuotationTable: SalesQuotationTable::write
Form SalesTable.modified
Form SalesTable.SalesTable_DS.linkActive
Form SalesTable.write
Form SalesTable: SalesLine::write
Form SalesTable: SalesTable::write
Form TMSRateRouteWorkbench.updateRoutes
Form VendEditInvoice: VendInvoiceInfoTable.write
Form WhsWorkTable.setFilter
FormLetterJournalPost.post
FormLetterService.run
Forms WHSLoadPlanningWorkbench.init
Forms WHSLoadPlanningWorkbench.restoreQuery
FreeTextInvoiceDP::insertIntoFreeTextInvoiceHeaderFooterTmp
From ProjTableCreate.init
HierarchyCreate.run
HierarchyTemplateCopyingDialog_proj.main
<table>
<thead>
<tr>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>InterCompanyPost.formLetterCollect</td>
</tr>
<tr>
<td>InventAgeDimDP.calcAllDim</td>
</tr>
<tr>
<td>InventAgeDimDP.insertInventAgeDimTmp</td>
</tr>
<tr>
<td>InventAgeDimDP.insertOrMergeInventAgeDimTmp</td>
</tr>
<tr>
<td>InventCountCreate.dialog</td>
</tr>
<tr>
<td>InventDimCtrl_Frm_Lookup.initDisplayOrderDataSource</td>
</tr>
<tr>
<td>InventDimPhysDP.processReport</td>
</tr>
<tr>
<td>InventDimViewContract</td>
</tr>
<tr>
<td>InventMovement.updateSerialNumIssue</td>
</tr>
<tr>
<td>InventMovement.updateSerialNumReceipt</td>
</tr>
<tr>
<td>InventMovement::updateLedgerPhysical</td>
</tr>
<tr>
<td>InventOnhandReserve.updateReserveNow</td>
</tr>
<tr>
<td>InventSumDateEngine.clearNotSelectedDimensions</td>
</tr>
<tr>
<td>InventTransferMulti.insert</td>
</tr>
<tr>
<td>InventUpd_Picked.updatePickMore</td>
</tr>
<tr>
<td>InventUpd_Reservation.updateReserveLess</td>
</tr>
<tr>
<td>InventUpdateOnhand.checkOnHand</td>
</tr>
<tr>
<td>InventValueReportContract</td>
</tr>
<tr>
<td>InventValueReportController</td>
</tr>
<tr>
<td>InventValueReportPopulateResource.initReportLines</td>
</tr>
<tr>
<td>JmgPostStandardSystem.postProjTime</td>
</tr>
<tr>
<td>JournalFormTable.designLookupJournalName</td>
</tr>
<tr>
<td>JournalFormTable.initAllOpenPostedFromCaller</td>
</tr>
<tr>
<td>LedgerBalancesBase.CalculateBalance</td>
</tr>
<tr>
<td>LedgerInAccountStatement.main</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>LedgerJournalCheckPost.createReverseEntryJournalLine</td>
</tr>
<tr>
<td>LedgerJournalCheckPost.postJournal</td>
</tr>
<tr>
<td>LedgerJournalCheckPost.runInternal</td>
</tr>
<tr>
<td>LedgerJournalCheckPost::updateSystemBlockCheckedPostedJournal</td>
</tr>
<tr>
<td>LedgerJournalMultiPost.multiSelectPost</td>
</tr>
<tr>
<td>LedgerJournalTrans table.checkBankAccounts</td>
</tr>
<tr>
<td>LedgerJournalTransUpdateVend::postNewVendorVoucher</td>
</tr>
<tr>
<td>Map ProjTableWizardCtrl::insertDB</td>
</tr>
<tr>
<td>Map SalesPurchLine.calcPrice2LineAmount</td>
</tr>
<tr>
<td>Map SalesPurchLine.resetPriceAgreement</td>
</tr>
<tr>
<td>Map SalesPurchLine.setPriceAgreement</td>
</tr>
<tr>
<td>MarkupAllocation.sumValue</td>
</tr>
<tr>
<td>MCRInventSearch.executeSearch</td>
</tr>
<tr>
<td>MCROrderEventTable.Insert</td>
</tr>
<tr>
<td>McrPriceHistoryForm.insertPriceHistory</td>
</tr>
<tr>
<td>PmfFormCtrl.initPost</td>
</tr>
<tr>
<td>PriceDiscAdmCheckPost.checkForOverlapsAndGaps</td>
</tr>
<tr>
<td>PriceDiscAdmCopy.updateNow</td>
</tr>
<tr>
<td>ProdJournalCheckPostProd::checkTrans</td>
</tr>
<tr>
<td>ProdJournalCheckPostProd::postTransLedger</td>
</tr>
<tr>
<td>ProdJournalCheckPostProd::postVoucher</td>
</tr>
<tr>
<td>ProdJournalCheckPostRoute.updateProdRouteScheduling</td>
</tr>
<tr>
<td>ProdJournalCreateProd.createLines</td>
</tr>
<tr>
<td>ProdJournalCreateRoute.createLinesProdRoute</td>
</tr>
<tr>
<td>ProdJournalFormTable.datasourceExecuteQueryPre</td>
</tr>
</tbody>
</table>
ProdMultiBOMCalc.run
ProdMultiCostEstimation.run
ProdMultiHistoricalCost.run
ProdMultiRelease.insert
ProdMultiRelease.run
ProdMultiReportFinished.main
ProdMultiReportFinished.run
ProdMultiSchedulingJob.run
ProdMultiSchedulingOperation.run
ProdMultiStartUp.run
ProdPurch.createPurchTable
prodTableChangeQtySched.performActionFromDefaultValues
prodTableChangeQtySched.performActionFromPrompt
ProdUpdCostEstimation.costEstimateOperations
ProdUpdCostEstimation.createPurchLine
ProdUpdReportFinished.run
ProdUpdReportFinished.updateBOMConsumption
ProdUpdStartUp.updateBOMConsumption
ProdUpdStartUp.updateRouteConsumption
ProjAdjustmentSelect.doTrans
ProjAdjustmentSelect.newQuery
ProjAdjustmentSelect.Run
ProjAdjustmentUpdate.checkTransChanged
ProjBudgetTransactionsManager.adjustBudget
ProjCopyForecastItem.copyToSalesLine
<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProjCostControl.createActualCosts</td>
</tr>
<tr>
<td>ProjCostControl.createActuals</td>
</tr>
<tr>
<td>ProjCostControl.createAverageForRemaining</td>
</tr>
<tr>
<td>ProjCostControl.createCommittedCosts</td>
</tr>
<tr>
<td>ProjCostControl.createForecastCosts</td>
</tr>
<tr>
<td>ProjCostControl.queryCommittedCosts</td>
</tr>
<tr>
<td>ProjCostControl.queryProjTransPosting</td>
</tr>
<tr>
<td>ProjCostControl.run</td>
</tr>
<tr>
<td>ProjCostControl.validate</td>
</tr>
<tr>
<td>ProjForecastBudgetCopy.do_cost</td>
</tr>
<tr>
<td>ProjForecastBudgetCopy.do_empl</td>
</tr>
<tr>
<td>ProjForecastBudgetCopy.do_OnaCC</td>
</tr>
<tr>
<td>ProjForecastBudgetCopy.do_Revenue</td>
</tr>
<tr>
<td>ProjForecastBudgetCopy.do_Sales</td>
</tr>
<tr>
<td>ProjForecastBudgetCopy.initQuery</td>
</tr>
<tr>
<td>ProjForecastBudgetCopy.validate</td>
</tr>
<tr>
<td>ProjForecastBudgetDelete.initQuery</td>
</tr>
<tr>
<td>ProjForecastTransferFromWbs.transferItemToForecast</td>
</tr>
<tr>
<td>ProjFundingEngine.allocate</td>
</tr>
<tr>
<td>ProjFundingEngine.isAmountWithinFundingLimits</td>
</tr>
<tr>
<td>ProjFundingEngine.updateFundingLimits</td>
</tr>
<tr>
<td>ProjInvoiceChooseNormal.dialog</td>
</tr>
<tr>
<td>ProjInvoiceJournalCreate.initTotals</td>
</tr>
<tr>
<td>ProjInvoiceJournalPost.createProjInvoiceCost</td>
</tr>
<tr>
<td>ProjInvoiceJournalPost.createProjInvoiceEmpl</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>ProjInvoiceJournalPost.createProjInvoiceItem</td>
</tr>
<tr>
<td>ProjInvoiceJournalPost.createProjInvoiceOnAcc</td>
</tr>
<tr>
<td>ProjInvoiceJournalPost.createProjInvoiceRevenue</td>
</tr>
<tr>
<td>ProjInvoiceJournalPost.postCustVend</td>
</tr>
<tr>
<td>ProjInvoiceProposalCreateLines.runSalesLineQuery</td>
</tr>
<tr>
<td>ProjInvoiceProposalCreateLines.runTransactions</td>
</tr>
<tr>
<td>ProjInvoiceProposalInsertLines.run</td>
</tr>
<tr>
<td>ProjInvoiceProposalNormalPeriodic.createParameters</td>
</tr>
<tr>
<td>ProjInvoiceProposalPeriodic.dialog</td>
</tr>
<tr>
<td>ProjJournalCheckPost.processHourJournalResourceRateCost</td>
</tr>
<tr>
<td>ProjLedger.initFromProjectPostingTransaction</td>
</tr>
<tr>
<td>ProjLedgerUpdate.insert</td>
</tr>
<tr>
<td>ProjPlanVersionsManager.copyTasks</td>
</tr>
<tr>
<td>ProjProposalTotals.calc</td>
</tr>
<tr>
<td>ProjSplitBill.buildRuleQR</td>
</tr>
<tr>
<td>ProjSplitBill.split</td>
</tr>
<tr>
<td>ProjStatisticCalc.mapPSAEntityToTmpProjStatistic</td>
</tr>
<tr>
<td>ProjValCheckTrans.setVariablesFromBuffer</td>
</tr>
<tr>
<td>PsaCustomerRetention.createFeeTransaction</td>
</tr>
<tr>
<td>PsaGenerateQuotationLines.createSalesQuotationLines</td>
</tr>
<tr>
<td>PsaProjInvoiceDP::insertPSAProjInvoiceHeaderTmp</td>
</tr>
<tr>
<td>PsaProjInvoiceDP::insertPSAProjInvoiceTmp</td>
</tr>
<tr>
<td>PSARetentionRelease.insertLineRecords</td>
</tr>
<tr>
<td>PurchAgreementGenerateReleaseOrder.check</td>
</tr>
<tr>
<td>PurchAutoCreate_RFQ.createPurchLine</td>
</tr>
<tr>
<td>METHOD</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>PurchAutoCreate_Sales.createLine</td>
</tr>
<tr>
<td>PurchCancel.cancelMarkup</td>
</tr>
<tr>
<td>PurchCancel.run</td>
</tr>
<tr>
<td>PurchCopying.copyLine</td>
</tr>
<tr>
<td>PurchCreateFromSalesOrder.mcrDropChipCreateTmpFrmVirtual</td>
</tr>
<tr>
<td>PurchCreateFromSalesOrder.preMatchIncludedLinesWithAgreements</td>
</tr>
<tr>
<td>PurchFormLetter::PrePromptInit</td>
</tr>
<tr>
<td>PurchFormLetter::Main</td>
</tr>
<tr>
<td>PurchFormLetter::MainOnServer</td>
</tr>
<tr>
<td>PurchFormletterParmData.createParmTable</td>
</tr>
<tr>
<td>PurchFormletterParmDataInvoice.chooseLinesFromPurchSelectLinesManager</td>
</tr>
<tr>
<td>PurchLineType.intercompanyMirror</td>
</tr>
<tr>
<td>PurchLineType_Project.initFromInventTable</td>
</tr>
<tr>
<td>PurchLineType_WithMultipleDeliveries.recalculateDeliveryScheduleOrderLine</td>
</tr>
<tr>
<td>PurchPackingSlipDP::setPurchPackingSlipDetailsTmp</td>
</tr>
<tr>
<td>PurchPackingSlipDP::setPurchPackingSlipHeaderTmp</td>
</tr>
<tr>
<td>PurchPurchaseOrderDP.createData</td>
</tr>
<tr>
<td>PurchPurchaseOrderDP.initializePurchPurchaseOrderHeader</td>
</tr>
<tr>
<td>PurchPurchaseOrderDP::setPurchPurchaseOrderDetails</td>
</tr>
<tr>
<td>PurchPurchaseOrderDP::setPurchPurchaseOrderHeader</td>
</tr>
<tr>
<td>PurchReceiptsListDP::setPurchReceiptsListDetailsTmp</td>
</tr>
<tr>
<td>PurchReceiptsListDP::setPurchReceiptsListHeaderTmp</td>
</tr>
<tr>
<td>PurchReqTable2LineField.lineUpdateDescription</td>
</tr>
<tr>
<td>METHOD</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>PurchRFQCaseAutoCreate.newAutoCreate</td>
</tr>
<tr>
<td>PurchRFQCompare.BuildReplyLineList</td>
</tr>
<tr>
<td>PurchRFQSendDP::processReport</td>
</tr>
<tr>
<td>PurchRFQSendJournalCreate.createOrUpdateRFQ</td>
</tr>
<tr>
<td>PurchTable2LineField.getFieldDescription</td>
</tr>
<tr>
<td>PurchTableType.intercompanyMirror</td>
</tr>
<tr>
<td>ReqActionApplyPurchaseOrder.applyActionToReferencedOrder</td>
</tr>
<tr>
<td>ReqBOMCreate.createBOM</td>
</tr>
<tr>
<td>ReqCalc.mcrInsertItemContinuitySales</td>
</tr>
<tr>
<td>ReqCalcScheduleItemTable.run</td>
</tr>
<tr>
<td>ReqSetupDim.setReqItemTableGrouped</td>
</tr>
<tr>
<td>ReqSupplyDemandScheduleModel.executeQuery</td>
</tr>
<tr>
<td>ReqSupplyDemandScheduleModel.insertPeriodValue</td>
</tr>
<tr>
<td>ReqTransPoMarkChangeToRFQ.change2RFQ</td>
</tr>
<tr>
<td>ReqTransPoMarkFirm.create</td>
</tr>
<tr>
<td>ReqTransPoMarkFirm.createInventTransferLine</td>
</tr>
<tr>
<td>ReqTransPoMarkFirm.createPurchLine</td>
</tr>
<tr>
<td>ReqTransPoMarkFirm.createPurchTable</td>
</tr>
<tr>
<td>ReqTransPoMarkFirm.firmSelectedPlannedOrders</td>
</tr>
<tr>
<td>RouteCopyToProd.copyTo</td>
</tr>
<tr>
<td>SalesAgreementGenerateReleaseOrder.check</td>
</tr>
<tr>
<td>SalesAgreementGenerateReleaseOrder.main</td>
</tr>
<tr>
<td>SalesAutoCreate_ReleaseFromAgreement.createSalesLine</td>
</tr>
<tr>
<td>SalesAutoCreate_ReleaseFromAgreement.createSalesTable</td>
</tr>
<tr>
<td>SalesAutoCreate_ReleaseOrder.createSalesTable</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>SalesConfirmDP::setSalesConfirmDetailsTmp</td>
</tr>
<tr>
<td>SalesConfirmDP::setSalesConfirmHeaderTmp</td>
</tr>
<tr>
<td>SalesCopying.copy</td>
</tr>
<tr>
<td>SalesCopying.copyHeader</td>
</tr>
<tr>
<td>SalesCopying.deleteLines</td>
</tr>
<tr>
<td>SalesCopying_CreditNote.copy</td>
</tr>
<tr>
<td>SalesCopying_CreditNote.copyHeader</td>
</tr>
<tr>
<td>SalesFormLettermainOnServer</td>
</tr>
<tr>
<td>SalesFormLetterreselect</td>
</tr>
<tr>
<td>SalesFormletterParmData.createParmLine</td>
</tr>
<tr>
<td>SalesFormletterParmData::createParmTable</td>
</tr>
<tr>
<td>SalesInvoiceController::outputReport</td>
</tr>
<tr>
<td>SalesInvoiceDP::useExistingReportData</td>
</tr>
<tr>
<td>SalesInvoiceDP::insertIntoSalesInvoiceHeaderFooterTmp</td>
</tr>
<tr>
<td>SalesInvoiceDP::insertIntoSalesInvoiceTmp</td>
</tr>
<tr>
<td>SalesLineExplodeBOM::explode</td>
</tr>
<tr>
<td>SalesLineType.canPickingListCanBeRegistered</td>
</tr>
<tr>
<td>SalesLineType.delete</td>
</tr>
<tr>
<td>SalesLineType.initDimensionsSpecificDefaulting</td>
</tr>
<tr>
<td>SalesLineType.initFromSalesLine</td>
</tr>
<tr>
<td>SalesLineType.interCompanyMirror</td>
</tr>
<tr>
<td>SalesLineType.setSalesStatus</td>
</tr>
<tr>
<td>SalesLineType.validateField field ShippingDateRequested and ShippingDateConfirmed</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>SalesPackingSlipDP::printDimHistory</td>
</tr>
<tr>
<td>SalesPackingSlipDP::setSalesPackingSlipDetailsTmp</td>
</tr>
<tr>
<td>SalesPackingSlipDP::setSalesPackingSlipHeaderTmp</td>
</tr>
<tr>
<td>SalesPurchTableToLineUpdate.update</td>
</tr>
<tr>
<td>SalesQuantity_PackingSlip.calcQtySales</td>
</tr>
<tr>
<td>SalesQuantity_PickingList.calcQtySales</td>
</tr>
<tr>
<td>SalesQuotationConfirmationDP::setSalesQuotationDetailsTmp</td>
</tr>
<tr>
<td>SalesQuotationConfirmationDP::setSalesQuotationHeaderTmp</td>
</tr>
<tr>
<td>SalesQuotationCopying.buildTreeControl</td>
</tr>
<tr>
<td>SalesQuotationCopying.Copy</td>
</tr>
<tr>
<td>SalesQuotationDP::setSalesQuotationDetailsTmp</td>
</tr>
<tr>
<td>SalesQuotationDP::setSalesQuotationHeaderTmp</td>
</tr>
<tr>
<td>SalesQuotationEditLinesForm.mainOnServer</td>
</tr>
<tr>
<td>SalesQuotationEditLinesForm_Proj_Confirm.queryBuildSalesQuotationTable</td>
</tr>
<tr>
<td>SalesQuotationEditLinesForm_Proj_Send.queryBuildSalesQuotationTable</td>
</tr>
<tr>
<td>SalesQuotationEditLinesForm_Sales_Confir.updateNow</td>
</tr>
<tr>
<td>SalesQuotationEditLinesForm_Sales_Confirm.createSalesLine</td>
</tr>
<tr>
<td>SalesQuotationEditLinesForm_Sales_Send.checkLines</td>
</tr>
<tr>
<td>SalesQuotationJumpRef.main</td>
</tr>
<tr>
<td>SalesQuotationLineType.initFromSalesQuotationLine</td>
</tr>
<tr>
<td>SalesQuotationLineType_Proj.validateWrite</td>
</tr>
<tr>
<td>SalesQuotationProjLinkWizard.transferForecastToProject</td>
</tr>
<tr>
<td>SalesQuotationProjLinkWizard.transferItemReq</td>
</tr>
<tr>
<td>SalesQuotationTransferToProject.transferItemsToForecast</td>
</tr>
<tr>
<td>SalesQuotationTransferToProject.transferItemsToItemReq</td>
</tr>
<tr>
<td>Method</td>
</tr>
<tr>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>SalesQuotationUpdate.getCallerModuleFromParm</td>
</tr>
<tr>
<td>SalesTableForm.initValues</td>
</tr>
<tr>
<td>SalesTableForm_DeliverySchedule.updateSalesLineTable</td>
</tr>
<tr>
<td>SalesTableType.intercompanyMirror</td>
</tr>
<tr>
<td>SalesTableType.update</td>
</tr>
<tr>
<td>SalesTableType.validateDelete</td>
</tr>
<tr>
<td>smmSalesCustItemStatisticsDP::processReport</td>
</tr>
<tr>
<td>Table CaseDetailBase.validateWrite</td>
</tr>
<tr>
<td>Table CustCollectionLetterJour.updateCollectionLetterCodeCustTrans()</td>
</tr>
<tr>
<td>Table EcoResProductTranslation.queryAddCompanyLanguage</td>
</tr>
<tr>
<td>Table InventLocation::lookupBySiteIdAllTypes</td>
</tr>
<tr>
<td>Table InventPosting.accountGroup</td>
</tr>
<tr>
<td>Table InventPosting.accountItemLedgerDimension</td>
</tr>
<tr>
<td>Table InventPosting.deleteFromCust</td>
</tr>
<tr>
<td>Table InventPosting.deleteFromVend</td>
</tr>
<tr>
<td>Table InventTable.defaultProductDescription</td>
</tr>
<tr>
<td>Table InventTable.defaultProductName</td>
</tr>
<tr>
<td>Table InventTable.lookupBOMId</td>
</tr>
<tr>
<td>Table InventTrans.updateMarkReqTransCov</td>
</tr>
<tr>
<td>Table PaymTerm.due</td>
</tr>
<tr>
<td>Table ProdBOM.updateStartUp</td>
</tr>
<tr>
<td>Table ProdBOM.updateSubPurch</td>
</tr>
<tr>
<td>Table ProdJournalBOM.insertJournalCreate</td>
</tr>
<tr>
<td>Table ProdJournalBOM.lookupTransId</td>
</tr>
<tr>
<td>Table ProdTable.validateRouteId</td>
</tr>
</tbody>
</table>
Table ProjBegBalJournalTrans_CostSales.postProjTransactionCost
Table ProjBegBalJournalTrans_CostSales.postProjTransactionHour
Table ProjBegBalJournalTrans_CostSales.postProjTransactionItem
Table ProjBegBalJournalTrans_Fee.postProjTransaction
Table ProjBegBalJournalTrans_OnAcc.postProjTransactionCost
Table PurchLine.initBarCode
Table PurchLine.priceDateDelegate
Table PurchLine.setPriceDisc
Table PurchTable.updateFromPurchReqLineMap
Table ReqPO.updateBOMRoute
Table ReqTrans.bulkInitFromInventTransOrigin
Table RouteOpr.validateFieldValue
Table RouteVersion.checkExistInventSiteId
Table SalesLine.checkPriceDate
Table Salesline.convertCurrencyCode
Table SalesLine.convertToDeliverySchedule
Table SalesLine.createFromTmpFrmVirtualIL
Table SalesLine.createReplacement
Table SalesLine.createSalesLine
Table SalesLine.expandBOM
Table Salesline.modifyInventDimSet
Table SalesLine.priceDateDelegate
Table salesLine.setPriceAgreement
Table Salesline.splitReturnLine
Table SalesLine::createFromSalesQuotationLine
<table>
<thead>
<tr>
<th>TABLE</th>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table SalesQuotationLine.createFromTmpFrmVirtual</td>
<td></td>
</tr>
<tr>
<td>Table SalesQuotationLine.modifiedField</td>
<td></td>
</tr>
<tr>
<td>Table SalesQuotationLine.modifyInventDim</td>
<td></td>
</tr>
<tr>
<td>Table SalesQuotationTable.copyAddressToLine</td>
<td></td>
</tr>
<tr>
<td>Table SalesQuotationTable.lookupTemplateName</td>
<td></td>
</tr>
<tr>
<td>Table SalesTable.copyAddressToLine</td>
<td></td>
</tr>
<tr>
<td>Table SalesTable.copyRMALines</td>
<td></td>
</tr>
<tr>
<td>Table SalesTable.copyThirdPartyBillingAddressToLine</td>
<td></td>
</tr>
<tr>
<td>Table SalesTable.existingJournals</td>
<td></td>
</tr>
<tr>
<td>Table SalesTable.initFromCustTableIL</td>
<td></td>
</tr>
<tr>
<td>Table SalesTable.initFromProjTable</td>
<td></td>
</tr>
<tr>
<td>Table SalesTable.initFromSalesQuotationTable</td>
<td></td>
</tr>
<tr>
<td>Table SalesTable.unlinkAgreement</td>
<td></td>
</tr>
<tr>
<td>Table WHSAccountItemStatusDefault.checkModuleAccountNum</td>
<td></td>
</tr>
<tr>
<td>Table WHSLoadLine.delete</td>
<td></td>
</tr>
<tr>
<td>Table WHSLoadLine.updateReleaseQty</td>
<td></td>
</tr>
<tr>
<td>Table WhsLoadLine.validateQty</td>
<td></td>
</tr>
<tr>
<td>Table WHSLoadTable.assignOriginInfo</td>
<td></td>
</tr>
<tr>
<td>Table WHSProdTable.pickMore</td>
<td></td>
</tr>
<tr>
<td>Table WhsWorkTable.lockUnLockWork</td>
<td></td>
</tr>
<tr>
<td>Table WMSBillOfLading.constructFromInvoice</td>
<td></td>
</tr>
<tr>
<td>Table WMSBillOfLading.constructFromPackingSlip</td>
<td></td>
</tr>
<tr>
<td>Table WMSBillOfLading.constructFromShipment</td>
<td></td>
</tr>
<tr>
<td>Table WMSOrderTrans.loopWMSOrderTransMulti</td>
<td></td>
</tr>
<tr>
<td>Table WrkCtrActivityRequirementSet.copyRequirements</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Table WrkCtrActivityRequirementSet.schedulingProperties</td>
<td></td>
</tr>
<tr>
<td>Tables SalesLine/Methods/setPriceDisc</td>
<td></td>
</tr>
<tr>
<td>TamDeductionUpdate_Deny.update</td>
<td></td>
</tr>
<tr>
<td>TmsProcessXML_Container.readShipContainer</td>
<td></td>
</tr>
<tr>
<td>TmsProcessXML_Shipment.readShipContainer</td>
<td></td>
</tr>
<tr>
<td>TradeInterCompany.insertInterCompanyInventDim</td>
<td></td>
</tr>
<tr>
<td>TradeInterCompanyConv.axPurchItemId</td>
<td></td>
</tr>
<tr>
<td>TradeInterCompanyConv.axSalesItemId</td>
<td></td>
</tr>
<tr>
<td>TransactionReversal_Asset.reversalBook</td>
<td></td>
</tr>
<tr>
<td>TransactionReversal_Cust::reversal</td>
<td></td>
</tr>
<tr>
<td>TransactionReversal_CustVend.createCustVendTrans</td>
<td></td>
</tr>
<tr>
<td>VendInvoiceDocumentDP::insertVendInvoiceDocumentTmp</td>
<td></td>
</tr>
<tr>
<td>VendInvoiceTableToLineUpdate.convertPurchTableFieldToVendInvoice</td>
<td></td>
</tr>
<tr>
<td>VendorInvoiceLineSourceDocLineItem.calculateSourceDocumentAmountMap</td>
<td></td>
</tr>
<tr>
<td>VersioningDocument.change</td>
<td></td>
</tr>
<tr>
<td>VersioningPurchaseOrder.createChangeRequest</td>
<td></td>
</tr>
<tr>
<td>WhsCycleCountCreateThreshold.processCycleCountThresholdItem</td>
<td></td>
</tr>
<tr>
<td>WHSInventOnHandReserve.changeReservation</td>
<td></td>
</tr>
<tr>
<td>WHSLaborStandards.findLaborStandardByItem</td>
<td></td>
</tr>
<tr>
<td>WHSLoadLine.update</td>
<td></td>
</tr>
<tr>
<td>WHSLoadTable.tmsLoadConfirmation</td>
<td></td>
</tr>
<tr>
<td>WHSLocationBuild</td>
<td></td>
</tr>
<tr>
<td>WHSLocationDirective.findLocation</td>
<td></td>
</tr>
<tr>
<td>WHSLocationDirective.findPickPutLocation</td>
<td></td>
</tr>
<tr>
<td>WHSLocationDirectiveActionQuery.modifyPickLocDirActionQuery</td>
<td></td>
</tr>
<tr>
<td>METHOD</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>WHSPool.pickFromWorkCenter</td>
<td></td>
</tr>
<tr>
<td>WHSPostEngineBase.prodCreateWork</td>
<td></td>
</tr>
<tr>
<td>WHSPostEngineBase.prodPickQty</td>
<td></td>
</tr>
<tr>
<td>WhsRfControlData.getClusterPickQty</td>
<td></td>
</tr>
<tr>
<td>WHSRFCControlData.processControl</td>
<td></td>
</tr>
<tr>
<td>WhsShipConfirm.canShipConfirm</td>
<td></td>
</tr>
<tr>
<td>WHSShipConfirm.createInventTransferParmLineFromContainerTable</td>
<td></td>
</tr>
<tr>
<td>WHSShipConfirm.runTransferShip</td>
<td></td>
</tr>
<tr>
<td>WhsShipConfirm.validateAllAllowedForOverOrUnderdeliveryWorkQtyHasBeenPicked</td>
<td></td>
</tr>
<tr>
<td>WHSSplitWork.splitWork</td>
<td></td>
</tr>
<tr>
<td>WHSWorkClusterTable.cleanupCluster</td>
<td></td>
</tr>
<tr>
<td>WHSWorkCreateProdPut.insertProdParmForCoByProduct</td>
<td></td>
</tr>
<tr>
<td>WHSWorkCreateProdPut.insertProdParmForProdItem</td>
<td></td>
</tr>
<tr>
<td>WhsWorkExecute.getFirstOpenLineSystemDirected</td>
<td></td>
</tr>
<tr>
<td>WHSWorkExecute.overPickByItem</td>
<td></td>
</tr>
<tr>
<td>WHSWorkExecute.putAwayToLocation</td>
<td></td>
</tr>
<tr>
<td>WHSWorkExecute.scanLicensePlate</td>
<td></td>
</tr>
<tr>
<td>WHSWorkExecuteDisplay.buildPORecTrackingDimensions</td>
<td></td>
</tr>
<tr>
<td>WhsWorkExecuteDisplayCycleCount.findOrCreateCycleCountWorkLines</td>
<td></td>
</tr>
<tr>
<td>WHSWorkExecuteDisplayLPReceiving.displayForm</td>
<td></td>
</tr>
<tr>
<td>WHSWorkExecuteDisplayMixedLPReceiving.displayForm</td>
<td></td>
</tr>
<tr>
<td>WHSWorkLine.cancelLineMultiPick</td>
<td></td>
</tr>
<tr>
<td>WHSWorkLine.cancelLinePartial</td>
<td></td>
</tr>
<tr>
<td>WMSArrivalCreateJournal.createWMSJournalTrans</td>
<td></td>
</tr>
<tr>
<td>WMSArrivalCreateJournal.createWMSJournalTransFromTmp</td>
<td></td>
</tr>
<tr>
<td>METHOD</td>
<td>METHOD</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>WMSArrivalOverviewGeneration.buildReturnOrderFromSalesLine</td>
<td>WMSJournalCheckPostReception.checkReference</td>
</tr>
<tr>
<td>WMSJournalTransUpdateSerialId.dialog</td>
<td>WmsPickingList_OrderPickDP::insertIntoTempTable</td>
</tr>
<tr>
<td>WrkCrtScheduler.writeJobCapacityReservations</td>
<td>WrkCrtApplicableResourceQuery.query</td>
</tr>
<tr>
<td>WrkCtrlScheduler_Prod.loadJobsDetail</td>
<td>WrkCtrlScheduler_Prod.saveOrder</td>
</tr>
<tr>
<td>WrkCtrlScheduler_Proj.saveOrder</td>
<td>WrkCtrlScheduler_Proj.writeJobData</td>
</tr>
<tr>
<td>WrkCtrlReservedSum.find</td>
<td>WrkCrtScheduler.computeJobTimes</td>
</tr>
<tr>
<td>WrkCrtScheduler.insertWrkCrtCapResUsingInsertList</td>
<td>WrkCrtScheduler.writeJobCapacityReservations</td>
</tr>
<tr>
<td>WrkCrtScheduler.writeJobData</td>
<td></td>
</tr>
</tbody>
</table>

**SQL operations made extensible**

Application code with embedded SQL statements cannot be modified through extensions. Changes have been made to the standard application to enable extensibility in the methods listed in the following table. This has commonly been enabled by transforming embedded SQL statements into query objects that support extending how SQL statements are built in these methods.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustCollectionLetterCreate.updateAllExisting</td>
<td>CustCollectionLetterCreate.updateExisting</td>
</tr>
<tr>
<td>CustProvisionalBalanceDP.insertCustProvisionalBalanceTmp</td>
<td>CustProvisionalBalanceDP.processReport</td>
</tr>
<tr>
<td>CustSettlementPriorityProcessing.createTempData</td>
<td>DirPartyTable.getLocationFromRole</td>
</tr>
</tbody>
</table>
InventQualityOrderTable.createInventQualityOrderLines
InventTransIdSum::calcSum
InventTransIdSum_InventLocation::calcSum
InventTransReference::setRefTrans
Markup.insertMarkupTrans
Markup.mcrCopyForReturn
PriceDisc.findDisc
PriceDisc.findPriceAgreement
PurchFormLetterParmDataInvoice.createLineProject
ReqPlanCopy.copyReqTransAndReqTransCov
ReqPlanCopy.copyReqTransKeep
ReqPlanCopy.copyWrkCtrCapRes
ReqPlanCopy.copyWrkCtrCapResForReqPO
SalesCopying.deleteLines
SalesLineType::deliveredInTotal
SalesTableType.parmPickingListRegistrationEnumerable
SalesTableType_Sales.canPickingListBeUpdated
SalesUpdateRemain.canclRemainderOnOpenSalesLines
SubledgerJournalizer.createSummaryFromJournalAccountEntry
SubledgerJournalizer.loadAccountingDistributionTmpJournalize
SubledgerJournalizer.loadFinalizeSubledgerJournalTmpDetail
SubledgerJournalizer.loadStandardSubledgerLedgerJournalTmpDetail
SubledgerJournalizer.loadSubledgerJourTmpDetailWithRelieving
SubledgerJournalizer.recordSubledgerJourAccEntriesForRounding
SubledgerJournalizer.summarizeJourAccountEntryDetailForRound
Maps enabled for extensibility

New patterns have been introduced for maps implementation that will allow you to add field and methods by extensions. Details on how this is done is available in the documentation both with maps that are used as interfaces and for versioning implementations.

The following table lists the maps and related tables where changes have been applied for enabling extensibility.

Maps and tables

<table>
<thead>
<tr>
<th>Map CustVendInvoiceTrans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map SalesPurchLine extensions or inheritance</td>
</tr>
<tr>
<td>Map SalesPurchTable</td>
</tr>
<tr>
<td>Map VendDocumentLineMap</td>
</tr>
<tr>
<td>Table PurchLine mappings</td>
</tr>
<tr>
<td>Table PurchLineHistory mappings</td>
</tr>
</tbody>
</table>

Inventory dimensions

This release introduces a new model for adding inventory dimensions. In previous releases it was not practical to support customization for new inventory dimensions if that required extending every SQL statement that included inventory dimensions. Instead, we have added 10 inventory dimensions without any specific designated usage. Partner solutions will code through indirection models that hold their code, and other models are made for individual implementation projects that deploy one or more of the prefabricated inventory dimensions toward use in a partner solution. Documentation will be available on how to implement with
inventory dimensions under this model, and release of a sample app with a Flavor dimension will help you learn
about the new model. The new inventory dimension can be freely deployed and used as either product
dimensions or tracking dimensions.

The changes have led to changing multiple places across the application, including what is shown in the
following list.

<table>
<thead>
<tr>
<th>CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extensible Product Dimensions</td>
</tr>
<tr>
<td>Form WHSInventOnHandReserve.updateInventDimFixedControls</td>
</tr>
<tr>
<td>InterCompanyInventDim: Condition with throw</td>
</tr>
<tr>
<td>InventDimFieldsMap - Added field</td>
</tr>
<tr>
<td>Inventory dimension InventUpdateOnhand::checkOnHand</td>
</tr>
<tr>
<td>Inventory Dimensions - Table InventDim.create</td>
</tr>
<tr>
<td>InventTransferUpdShip.populateIssueReceiptDimensions</td>
</tr>
<tr>
<td>Map InventInventoryDimensionEntityFieldsMapping::resolveInventDim()</td>
</tr>
<tr>
<td>Rename InventDimFieldsMap::getFieldIdForDimensionOnMappedTable to inventoryDimensionFieldIdOnMappedTable()</td>
</tr>
<tr>
<td>Table BOMConsistOfTmp mappings</td>
</tr>
<tr>
<td>Table BOMPartOfTmp mappings</td>
</tr>
<tr>
<td>Table EcoResTrackingDimensionGroup.isDimFieldTrackingDimension</td>
</tr>
<tr>
<td>Table InterCompanyInventDim mappings</td>
</tr>
<tr>
<td>Table InventAgeGroupDimTmp mappings</td>
</tr>
<tr>
<td>Table InventCheckReceiptCostPricePcsTmp mappings</td>
</tr>
<tr>
<td>Table InventCostTmpTransBreakdown mappings</td>
</tr>
<tr>
<td>Table InventCountStatisticsTmp mappings</td>
</tr>
<tr>
<td>Table InventDim mappings</td>
</tr>
<tr>
<td>Table InventOnhandTmp mappings</td>
</tr>
<tr>
<td>Table InventPhysicalPerWarehouseTransTmp_IT mappings</td>
</tr>
<tr>
<td>Table InventPriceOverviewTmp mappings</td>
</tr>
<tr>
<td>Table InventSumCriticalTmp mappings</td>
</tr>
</tbody>
</table>
CHANGE

Table InventSumDateTransReport mappings

Table InventSumDeltaDim mappings

Table InventTable mappings

Table InventTransferOrderOverviewTmp mappings

Table InventValueReportTmpLine mappings

Table ProdPickList mappings

Table SalesInvoiceTmp mappings

Table WHSPurchLine.registerPurchaseLine

Table WHSTmpCompleteWorkLine.lookupBatch

Table WHSTmpCompleteWorkLine.lookupTargetLicensePlateId

Table WMSCheckABCZonesTmp mappings

Table WMSPickingList_OrderPickTmp mappings

Table WMSPickingListReportTmp mappings

Metadata changes to enable extensibility

The following table lists changes made for enabling extensibility for specific metadata on these objects. These changes vary from instance to instance, you can consult the specific implementation to review the changes.

CHANGE

CountryRegionCodes property

CustCustomerEntity

EcoResProductCategoryAssignmentEntity made public

Form AssetSplit : FormControls

Form CustCollections.Cases

Form CustGroup

Form LedgerJournalTransCustPaym - menu item button auto declaration

Form LedgerJournalTransVendPaym - menu item button auto declaration

Table DimensionAttributeValueSetItem
Table EcoResReleasedProductCreationStaging missing ReplacementKey like other staging tables.

Tables SubledgerJournalAccountEntry(Tmp...)

View SubLedgerJournalAccountEntryView

Other changes

The following table lists additional changes that have been made for extensibility.

CHANGE

CustCollectionLetterCreate

CustCollectionLetterPost

Extensibility approach for number of decimal places for currency

Extensible edt decimal places: AssetDepreciationAmountUnit

Form Extension - DirPartyTable - registerOverrideMethod jumpRef

Form ProjCategoryLookup

Method signature changed: InventPostingSetupCache

Method signature changed: Table ProjCostPriceExpense.find

Method signature changed: Table ProjCostPriceExpense.findCostPrice

Method signature changed: Table ProjCostSalesPrice.find

Method signature changed: Table ProjCostSalesPrice.findCostSalesPrice

Method signature changed: Table ProjHourCostPrice.Find

Method signature changed: Table ProjHourCostPrice.FindCostPrice

Method signature changed: Table ProjHourSalesPrice.find

Method signature changed: Table ProjHourSalesPrice.findHourSalesPrice

New Quantity EDT added to ApplicationCommon

Other: New base enum for Price & Discount framework

Set Alternative Key = Yes to enable reference group lookups

Use of interface EcoResIProductCrossTableData
Bugs

The following table lists changes that were requested for extensibility but were acknowledged as bugs and fixed in the standard application.

<table>
<thead>
<tr>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>class CaseUpdateStatus_Close, method changeStatus</td>
</tr>
<tr>
<td>Incorrect relation on CustCollectionLetterJour</td>
</tr>
<tr>
<td>Other: Bug fix on CompanyHelpertestCreateParameter</td>
</tr>
<tr>
<td>Table CustCollectionLetterJour - class cancelCollectionLetterCodeCustTrans</td>
</tr>
</tbody>
</table>
This is a list of extensibility features that were implemented in the Dynamics 365 for Finance and Operations, Enterprise edition (July 2017). This version was released in July 2017 and has a build number of 7.2.11792.56024. For more information about the schedule of changes that support extensibility, see Application extensibility roadmap.

Soft-sealed application models

The following application middle-tier models were soft-sealed in this release. Overlayered code in these models will generate warnings on compilation.

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>MODEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Frameworks</td>
<td>CaseManagement</td>
</tr>
<tr>
<td>Application Frameworks</td>
<td>Dimensions</td>
</tr>
<tr>
<td>Application Frameworks</td>
<td>Directory</td>
</tr>
<tr>
<td>Application Frameworks</td>
<td>Organization</td>
</tr>
<tr>
<td>Application Frameworks</td>
<td>Currency</td>
</tr>
<tr>
<td>Application Frameworks</td>
<td>ApplicationCommon</td>
</tr>
<tr>
<td>HCM Core Models</td>
<td>3817938</td>
</tr>
<tr>
<td>Tax Models</td>
<td>Tax</td>
</tr>
<tr>
<td>Tax Models</td>
<td>Tax Books</td>
</tr>
<tr>
<td>Tax Models</td>
<td>Tax Books Application Suite Integration</td>
</tr>
<tr>
<td>Tax Models</td>
<td>Tax Engine Application Suite Integration</td>
</tr>
<tr>
<td>Tax Models</td>
<td>Tax Engine Configuration</td>
</tr>
<tr>
<td>Tax Models</td>
<td>Tax Engine Interface</td>
</tr>
<tr>
<td>Tax Models</td>
<td>Tax Engine Runtime Generation</td>
</tr>
<tr>
<td>Tax Models</td>
<td>TaxEngine</td>
</tr>
<tr>
<td>Source Document Models</td>
<td>SourceDocumentation</td>
</tr>
<tr>
<td>CATEGORY</td>
<td>MODEL</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>Source Document Models</td>
<td>SourceDocumentationTypes</td>
</tr>
<tr>
<td>Ledger Models</td>
<td>GeneralLedger</td>
</tr>
<tr>
<td>Ledger Models</td>
<td>Ledger</td>
</tr>
<tr>
<td>Ledger Models</td>
<td>Subledger</td>
</tr>
<tr>
<td>Middle Tier SCM Models</td>
<td>CostAccounting</td>
</tr>
<tr>
<td>Middle Tier SCM Models</td>
<td>CostAccountingAX</td>
</tr>
<tr>
<td>Middle Tier SCM Models</td>
<td>SCMControls</td>
</tr>
<tr>
<td>Middle Tier SCM Models</td>
<td>SCMMobile</td>
</tr>
<tr>
<td>Middle Tier SCM Models</td>
<td>UnitOfMeasure</td>
</tr>
<tr>
<td>Middle Tier SCM Models</td>
<td>WMSAdvancedMigration</td>
</tr>
<tr>
<td>Middle Tier SCM Models</td>
<td>InventoryDimensionConversion</td>
</tr>
<tr>
<td>Workspaces</td>
<td>ApplicationWorkspaces</td>
</tr>
<tr>
<td>Finance</td>
<td>Fiscalbooks</td>
</tr>
<tr>
<td>Management tools</td>
<td>DataUpgrade</td>
</tr>
</tbody>
</table>

**Hard-sealed application models**

The following application middle-tier models were hard-sealed in this release. Overlayered code in these models will generate errors on compilation.

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>MODEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accounts Payable</td>
<td>AccountsPayableMobile</td>
</tr>
<tr>
<td>Finance</td>
<td>FinancialReportingEntityStore</td>
</tr>
<tr>
<td>Tools</td>
<td>PerformanceTool</td>
</tr>
<tr>
<td>Expenses</td>
<td>ExpenseMobile</td>
</tr>
<tr>
<td>GER</td>
<td>ElectronicReporting</td>
</tr>
<tr>
<td>GER</td>
<td>ElectronicReportingCore</td>
</tr>
<tr>
<td>GER</td>
<td>Electronic Reporting Application Suite Integration</td>
</tr>
</tbody>
</table>
Enumerations that are now extensible

The following changes were made to support extending enumerations:

- Many enumerations in the standard application have been made extensible. An enumeration is made extensible by setting two properties on the enumeration. The `IsExtensible` property is set to Yes, and the `UseEnumValue` property is set to No.
- Some enumerations represent state. New façade methods have been added to help enable adding enumeration values by extension. For information about how to extend an enumeration, see Add values to enums through extension.
- Some application code that uses enumerations was changed to support extensibility. Common changes include:
  - Removing `throw` exception statements in the default case of a switch to allow post-event subscription.
  - Adding `SysExtension` support for extension.
  - Adding explicit delegates.

<table>
<thead>
<tr>
<th>Enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgreementState</td>
</tr>
<tr>
<td>AssetAccountType</td>
</tr>
<tr>
<td>AssetTransTypeJournal</td>
</tr>
<tr>
<td>BankAccountType</td>
</tr>
<tr>
<td>BankFormat</td>
</tr>
<tr>
<td>BarcodeContentType</td>
</tr>
<tr>
<td>BarcodeCoverPageEntityType</td>
</tr>
<tr>
<td>BarcodeType</td>
</tr>
<tr>
<td>BOMCalcCostingVersionUpdate</td>
</tr>
<tr>
<td>BOMCalcCostPriceUsed</td>
</tr>
<tr>
<td>BOMCalcSalesPriceUsed</td>
</tr>
<tr>
<td>BOMCalcType</td>
</tr>
<tr>
<td>BOMCheckLevel</td>
</tr>
<tr>
<td>BOMCopyContext</td>
</tr>
<tr>
<td>BOMCopyMethod</td>
</tr>
<tr>
<td>BOMCopyType</td>
</tr>
<tr>
<td>BOMCostCalculationMethod</td>
</tr>
</tbody>
</table>
### ENUMERATION

- BOMExplode
- BOMRouteCopyDataType
- BOMVersionFilter
- BudgetReservation_BusinessEvent_PSN
- BudgetReservation_SourceDocument_PSN
- BudgetReservation_SourceDocumentLine_PSN
- CatCallMethod
- CatContentType
- CatImportStatus
- CatMaintenanceRequestWFStatus
- CatProcurementErrorCode
- CatPurchaseStatus
- CatUserReviewApprovalStatus
- CatVendorCatalogFileUploadType
- CatVendorCatalogTemplateCategory
- CatVendorCategoryHierarchyType
- CatVendorConfigurationForImport
- CatVendorLegalEntityStatus
- CatVendorSiteType
- ConsignmentReplenishmentOrderLineStatus
- ConsignmentReplenishmentOrderStatus
- CostBreakdown
- CostCalculationCompareProductType
- CostCalculationRole
- CostCalculationState
<table>
<thead>
<tr>
<th>ENUMERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CostCalculationSurchargeSubtype</td>
</tr>
<tr>
<td>CostingActivationType</td>
</tr>
<tr>
<td>CostingVersionCompareTo</td>
</tr>
<tr>
<td>CostingVersionPriceType</td>
</tr>
<tr>
<td>CostPriceBase</td>
</tr>
<tr>
<td>CostProfitSet</td>
</tr>
<tr>
<td>CostSalesPriceDisplay</td>
</tr>
<tr>
<td>CostSheetNodeListType</td>
</tr>
<tr>
<td>CostSheetPanelView</td>
</tr>
<tr>
<td>CostSheetProdFlowMode</td>
</tr>
<tr>
<td>CostStatementCacheAggregationAfter</td>
</tr>
<tr>
<td>CostWIPStatementCategory</td>
</tr>
<tr>
<td>CustPaymentValidate</td>
</tr>
<tr>
<td>CustSpecTransOverviewFormMode</td>
</tr>
<tr>
<td>CustTransReType</td>
</tr>
<tr>
<td>CustVendPaymentStatus</td>
</tr>
<tr>
<td>CustVendTransportPointTypeTransfer</td>
</tr>
<tr>
<td>DlvScheduleMarkupConversionMode</td>
</tr>
<tr>
<td>EcoResAttributeModifier</td>
</tr>
<tr>
<td>EcoResCategoryAttributeModifier</td>
</tr>
<tr>
<td>EcoResCategoryChangeStatus</td>
</tr>
<tr>
<td>EcoResCategoryHierarchyModifier</td>
</tr>
<tr>
<td>EcoResCategoryNamedHierarchyRole</td>
</tr>
<tr>
<td>EcoResProductImageUsage</td>
</tr>
<tr>
<td>EcoResProductListPage</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ENUMERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>EcoResProductPerCompanyListPageType</td>
</tr>
<tr>
<td>EcoResProductTemplateType</td>
</tr>
<tr>
<td>EcoResReleaseProductToCompany</td>
</tr>
<tr>
<td>EcoResVariantConfigurationTechnologyType</td>
</tr>
<tr>
<td>ECPsalesOrdersViewType</td>
</tr>
<tr>
<td>EPCSSProductViewType</td>
</tr>
<tr>
<td>EUSalesTransMethod</td>
</tr>
<tr>
<td>FormLetterType</td>
</tr>
<tr>
<td>GanttCallerWrkCtr</td>
</tr>
<tr>
<td>GanttSetupType</td>
</tr>
<tr>
<td>GanttTimeUnit</td>
</tr>
<tr>
<td>GanttWrkCtrDisplayColumnsType</td>
</tr>
<tr>
<td>IntercompanyGoodsInTransitLineType</td>
</tr>
<tr>
<td>InterCompanyGoodsInTransitOrigin</td>
</tr>
<tr>
<td>InventAccountType</td>
</tr>
<tr>
<td>InventAccountTypeStdCostVariance</td>
</tr>
<tr>
<td>InventAdjustmentBy</td>
</tr>
<tr>
<td>InventAgingView</td>
</tr>
<tr>
<td>InventBatchJournalType</td>
</tr>
<tr>
<td>InventCostBundleState</td>
</tr>
<tr>
<td>InventCostCostDistribution</td>
</tr>
<tr>
<td>InventCostTransactionCategory</td>
</tr>
<tr>
<td>InventCostTransRefType</td>
</tr>
<tr>
<td>InventCountCode</td>
</tr>
<tr>
<td>InventItemCostingType</td>
</tr>
<tr>
<td>Enumeration</td>
</tr>
<tr>
<td>-----------------------------</td>
</tr>
<tr>
<td>InventItemLookupDefaultTab</td>
</tr>
<tr>
<td>InventItemOrderSetupCallerType</td>
</tr>
<tr>
<td>InventItemOrderSetupType</td>
</tr>
<tr>
<td>InventItemPriceCompareLevel</td>
</tr>
<tr>
<td>InventItemPriceFilterType</td>
</tr>
<tr>
<td>InventItemPriceType</td>
</tr>
<tr>
<td>InventJournalOwnershipChangeLineCreateQueryStatusIssue</td>
</tr>
<tr>
<td>InventJournalType</td>
</tr>
<tr>
<td>InventLedgerConflictModule</td>
</tr>
<tr>
<td>InventLocationType</td>
</tr>
<tr>
<td>InventMovSubType</td>
</tr>
<tr>
<td>InventNonConformanceApproval</td>
</tr>
<tr>
<td>InventNonConformanceHistoryType</td>
</tr>
<tr>
<td>InventNonConformanceType</td>
</tr>
<tr>
<td>InventParameters</td>
</tr>
<tr>
<td>InventPhysicalReduction</td>
</tr>
<tr>
<td>InventRefType</td>
</tr>
<tr>
<td>InventReleaseOrderPickingType</td>
</tr>
<tr>
<td>InventReportDimHistoryLogType</td>
</tr>
<tr>
<td>InventStdCostConvItemStatus</td>
</tr>
<tr>
<td>InventStdCostPeriodType</td>
</tr>
<tr>
<td>InventSumFields</td>
</tr>
<tr>
<td>InventSupplyDlvModeSelectCust</td>
</tr>
<tr>
<td>InventSupplyDlvModeSelectSupply</td>
</tr>
<tr>
<td>InventSupplyLeadTimeSource</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>InventSupplyTmpLeadtimeType</td>
</tr>
<tr>
<td>InventTestActionOnFailure</td>
</tr>
<tr>
<td>InventTestBlockProcess</td>
</tr>
<tr>
<td>InventTestCorrectionPriority</td>
</tr>
<tr>
<td>InventTestCorrectionStatus</td>
</tr>
<tr>
<td>InventTestDocumentStatus</td>
</tr>
<tr>
<td>InventTestOrderStatusDisplay</td>
</tr>
<tr>
<td>InventTestOutcomeStatus</td>
</tr>
<tr>
<td>InventTestQtySpecification</td>
</tr>
<tr>
<td>InventTestQuarantineType</td>
</tr>
<tr>
<td>InventTestReferenceType</td>
</tr>
<tr>
<td>InventTestReport</td>
</tr>
<tr>
<td>InventTestType</td>
</tr>
<tr>
<td>InventTrackingDimNodeType</td>
</tr>
<tr>
<td>InventTrackingProductType</td>
</tr>
<tr>
<td>InventTrackingRegisterTransRegStatus</td>
</tr>
<tr>
<td>InventTransChildType</td>
</tr>
<tr>
<td>InventTransferRemainStatus</td>
</tr>
<tr>
<td>InventTransferStatus</td>
</tr>
<tr>
<td>InventTransferUpdateType</td>
</tr>
<tr>
<td>InventTransPickRegisterLineStatus</td>
</tr>
<tr>
<td>InventTransType</td>
</tr>
<tr>
<td>InventUpdType</td>
</tr>
<tr>
<td>InventValueReportLedgerAccountCategory</td>
</tr>
<tr>
<td>InventValueReportLedgerLineType</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>-------------------------------------</td>
</tr>
<tr>
<td>InventValueReportResourceType</td>
</tr>
<tr>
<td>ItemGroupLedgerDimensionGroup</td>
</tr>
<tr>
<td>ItemReservation</td>
</tr>
<tr>
<td>JmgAbsenceColumnLayout</td>
</tr>
<tr>
<td>JmgAbsenceMethodEnum</td>
</tr>
<tr>
<td>JmgAttendanceRegistrationType</td>
</tr>
<tr>
<td>JmgAttendanceReportType</td>
</tr>
<tr>
<td>JmgBarCodeType</td>
</tr>
<tr>
<td>JmgBreakDropEnum</td>
</tr>
<tr>
<td>JmgClockStyle</td>
</tr>
<tr>
<td>JmgControlType</td>
</tr>
<tr>
<td>JmgDaysTotalWorkflowStatus</td>
</tr>
<tr>
<td>JmgEmployeeSignInStatus</td>
</tr>
<tr>
<td>JmgFeedbackButtonFunction</td>
</tr>
<tr>
<td>JmgFeedbackStyle</td>
</tr>
<tr>
<td>JmgFieldName</td>
</tr>
<tr>
<td>JmgGetRegistrationTimeFrom</td>
</tr>
<tr>
<td>JmgGridAppearance</td>
</tr>
<tr>
<td>JmgJobTableSynchronizationMode</td>
</tr>
<tr>
<td>JmgJournalRegWorkflowStatus</td>
</tr>
<tr>
<td>JmgMark</td>
</tr>
<tr>
<td>JmgMessageType</td>
</tr>
<tr>
<td>JmgPayAdjustType</td>
</tr>
<tr>
<td>JmgPayEventsExportType</td>
</tr>
<tr>
<td>JmgPaySpecTypeEnum</td>
</tr>
<tr>
<td>JmgPaySpecTypeEnum</td>
</tr>
<tr>
<td>Enumerations</td>
</tr>
<tr>
<td>------------------------------------</td>
</tr>
<tr>
<td>JmgPaySpecTypeEnumPick</td>
</tr>
<tr>
<td>JmgPostAutomatically</td>
</tr>
<tr>
<td>JmgProdStatusUpdate</td>
</tr>
<tr>
<td>JmgProdStatusUpdateReportFinished</td>
</tr>
<tr>
<td>JmgProfileSpecTypeEnum</td>
</tr>
<tr>
<td>JmgProfileStartCodeBlankPrev</td>
</tr>
<tr>
<td>JmgProjStatusUpdate</td>
</tr>
<tr>
<td>JmgRegistrationTouchJobStatus</td>
</tr>
<tr>
<td>JmgSecondPresentationEnum</td>
</tr>
<tr>
<td>JmgShopFloorServiceStatus</td>
</tr>
<tr>
<td>JmgSignInButtonFunction</td>
</tr>
<tr>
<td>JmgStoppedCompletedStatus</td>
</tr>
<tr>
<td>JmgTermBaudeRate</td>
</tr>
<tr>
<td>JmgTermComPort</td>
</tr>
<tr>
<td>JmgTermDataBit</td>
</tr>
<tr>
<td>JmgTerminalInsertMode</td>
</tr>
<tr>
<td>JmgTermStopBit</td>
</tr>
<tr>
<td>KanbanBoardRefreshCycle</td>
</tr>
<tr>
<td>KanbanBoardType</td>
</tr>
<tr>
<td>KanbanCardAssignmentType</td>
</tr>
<tr>
<td>KanbanControlActionState</td>
</tr>
<tr>
<td>KanbanControlLegendFormat</td>
</tr>
<tr>
<td>KanbanControlSelectionChanged</td>
</tr>
<tr>
<td>KanbanDemandOriginType</td>
</tr>
<tr>
<td>KanbanJobPeggingType</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>-----------------------------------</td>
</tr>
<tr>
<td>KanbanJobPickingListLineType</td>
</tr>
<tr>
<td>KanbanLineEventType</td>
</tr>
<tr>
<td>KanbanMultiMode</td>
</tr>
<tr>
<td>KanbanPrintInstructions</td>
</tr>
<tr>
<td>KanbanProdBOMLineEventType</td>
</tr>
<tr>
<td>KanbanQuantityCalculationStatus</td>
</tr>
<tr>
<td>KanbanSalesLineEventType</td>
</tr>
<tr>
<td>KanbanStockReplenishmentEventType</td>
</tr>
<tr>
<td>LeanBOMLineReservationMethod</td>
</tr>
<tr>
<td>LeanCostingUnusedQtyType</td>
</tr>
<tr>
<td>LeanHandlingUnitEmptyPolicy</td>
</tr>
<tr>
<td>LeanInventoryControl</td>
</tr>
<tr>
<td>LeanPeggedEventType</td>
</tr>
<tr>
<td>LeanPlanJobReferenceTypes</td>
</tr>
<tr>
<td>LeanProductionFlowCostingStatus</td>
</tr>
<tr>
<td>LeanProductionFlowVisualizationViewMode</td>
</tr>
<tr>
<td>LeanProductTypes</td>
</tr>
<tr>
<td>LeanTaktStatus</td>
</tr>
<tr>
<td>LedgerPostingType</td>
</tr>
<tr>
<td>LedgerTransTxt</td>
</tr>
<tr>
<td>MarkupAllocateAfter</td>
</tr>
<tr>
<td>MarkupCategory</td>
</tr>
<tr>
<td>MCRBrokerContractStatus</td>
</tr>
<tr>
<td>MCRCustSearchType</td>
</tr>
<tr>
<td>MCRFullTextSearchType</td>
</tr>
<tr>
<td>Enumeration</td>
</tr>
<tr>
<td>-----------------------------</td>
</tr>
<tr>
<td>MCRInstallPlanApplyMiscCharge</td>
</tr>
<tr>
<td>MCRItemListGenerationType</td>
</tr>
<tr>
<td>MCRPickingPrompt</td>
</tr>
<tr>
<td>MCRPickingSessionStatus</td>
</tr>
<tr>
<td>MCRPickingWaveStatus</td>
</tr>
<tr>
<td>MCRPriceHistoryType</td>
</tr>
<tr>
<td>MCRRoyaltyLineBreakType</td>
</tr>
<tr>
<td>MCRRoyaltyTakenFrom</td>
</tr>
<tr>
<td>MCRRoyaltyTransactionType</td>
</tr>
<tr>
<td>MCRRoyaltyUnitType</td>
</tr>
<tr>
<td>MCRRoyaltyUOMOption</td>
</tr>
<tr>
<td>MCRSalesOrderDetailStatus</td>
</tr>
<tr>
<td>ModuleInventCustVend</td>
</tr>
<tr>
<td>ModuleInventPurchSales</td>
</tr>
<tr>
<td>OriginalDocument</td>
</tr>
<tr>
<td>PaymDocumentType</td>
</tr>
<tr>
<td>PCExpressionEditorSymbolType</td>
</tr>
<tr>
<td>PCLookupMethod</td>
</tr>
<tr>
<td>PCNewSelectComponent</td>
</tr>
<tr>
<td>PCRrequirement</td>
</tr>
<tr>
<td>PCTableConstraintType</td>
</tr>
<tr>
<td>PDSAdjustmentPrinciple</td>
</tr>
<tr>
<td>PdsBatchAttribToleranceAction</td>
</tr>
<tr>
<td>PdsBatchAttribUpdateType</td>
</tr>
<tr>
<td>PDSCalcElementBase</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>PDSCompensationPrincipleEnum</td>
</tr>
<tr>
<td>PDSElementTypeEnum</td>
</tr>
<tr>
<td>PDSIngredientTypeEnum</td>
</tr>
<tr>
<td>PdsMRCDocumentStatus</td>
</tr>
<tr>
<td>PdsMRCEffectiveDateBasis</td>
</tr>
<tr>
<td>PdsMRCEventModule</td>
</tr>
<tr>
<td>PdsMRCEventType</td>
</tr>
<tr>
<td>PdsMRCListType</td>
</tr>
<tr>
<td>PdsPaymtType</td>
</tr>
<tr>
<td>PDSPotencyAttribRecordingEnum</td>
</tr>
<tr>
<td>PdsRebateCalcDateType</td>
</tr>
<tr>
<td>PdsRebateTakenFrom</td>
</tr>
<tr>
<td>PdsRebateUOMOption</td>
</tr>
<tr>
<td>PdsSameLotError</td>
</tr>
<tr>
<td>pdsTMAJournalPosting</td>
</tr>
<tr>
<td>PdsUpdateBatchDate</td>
</tr>
<tr>
<td>PdsUpdateDispositionStatus_Quality</td>
</tr>
<tr>
<td>PlanActivityCreateRelationType</td>
</tr>
<tr>
<td>PlanActivityProductionFlowActivityType</td>
</tr>
<tr>
<td>PlanTypes</td>
</tr>
<tr>
<td>PmfCostAllocationMethod</td>
</tr>
<tr>
<td>PmfOrderType</td>
</tr>
<tr>
<td>PmfOrderTypeFilter</td>
</tr>
<tr>
<td>PmfProdType</td>
</tr>
<tr>
<td>PMFSeqCalendarPeriod</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>PriceBase</td>
</tr>
<tr>
<td>PriceDiscPurchasePromptSystemSource</td>
</tr>
<tr>
<td>PriceDiscSalesPromptSystemSource</td>
</tr>
<tr>
<td>PriceGroupType</td>
</tr>
<tr>
<td>PriceSalesPurch</td>
</tr>
<tr>
<td>PriceType</td>
</tr>
<tr>
<td>ProcCategoryAdministrationActivity</td>
</tr>
<tr>
<td>ProdBOMConumpProposal</td>
</tr>
<tr>
<td>ProdBOMJournalQty</td>
</tr>
<tr>
<td>ProdBOMJournalSplit</td>
</tr>
<tr>
<td>ProdErrorCause</td>
</tr>
<tr>
<td>ProdGanttJobColorType</td>
</tr>
<tr>
<td>ProdGanttLoad</td>
</tr>
<tr>
<td>ProdGanttRouteColorType</td>
</tr>
<tr>
<td>ProdJournalCleanUpMode</td>
</tr>
<tr>
<td>ProdMode</td>
</tr>
<tr>
<td>ProdNotificationLevel</td>
</tr>
<tr>
<td>ProdParamInventDimLookup</td>
</tr>
<tr>
<td>ProdParmType</td>
</tr>
<tr>
<td>ProdRefLookUp</td>
</tr>
<tr>
<td>ProdRouteJobCurrentFormTabId</td>
</tr>
<tr>
<td>ProdSchedDirection</td>
</tr>
<tr>
<td>ProdScrapMethod</td>
</tr>
<tr>
<td>ProdStandardCostVariance</td>
</tr>
<tr>
<td>ProdStatusAll</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>---------------------------</td>
</tr>
<tr>
<td>ProdStatusType</td>
</tr>
<tr>
<td>ProductionTransType</td>
</tr>
<tr>
<td>ProdUpdateJour</td>
</tr>
<tr>
<td>ProdWHSReleasePolicy</td>
</tr>
<tr>
<td>ProdWIPType_NA</td>
</tr>
<tr>
<td>PurchaseOrderResponseAction</td>
</tr>
<tr>
<td>PurchaseType</td>
</tr>
<tr>
<td>PurchasingTransactionType</td>
</tr>
<tr>
<td>PurchCORReceivingMethod</td>
</tr>
<tr>
<td>PurchCORRejectStatus</td>
</tr>
<tr>
<td>PurchCovRef</td>
</tr>
<tr>
<td>PurchDlvAddr</td>
</tr>
<tr>
<td>PurchLineBackOrderViews</td>
</tr>
<tr>
<td>PurchLineDeliveryFulfillment</td>
</tr>
<tr>
<td>PurchLineDeliveryPrecision</td>
</tr>
<tr>
<td>PurchMatchingPolicyOverrideOption</td>
</tr>
<tr>
<td>PurchPrepayApplicationPolicy</td>
</tr>
<tr>
<td>PurchPriceDateType</td>
</tr>
<tr>
<td>PurchPurchaseOrderCreationMethod</td>
</tr>
<tr>
<td>PurchReApprovalPolicyRuleViewType</td>
</tr>
<tr>
<td>PurchReqAuthorizationSpecificReporting</td>
</tr>
<tr>
<td>PurchReqAutoCreatePurch</td>
</tr>
<tr>
<td>PurchReqCatalogAllNon</td>
</tr>
<tr>
<td>PurchReqConsolidationActiveStatus</td>
</tr>
<tr>
<td>PurchReqConsolidationPriority</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>PurchReqConsolidationStatus</td>
</tr>
<tr>
<td>PurchReqCreationStatus</td>
</tr>
<tr>
<td>PurchReqItemDescriptionTransfer</td>
</tr>
<tr>
<td>PurchReqItemFilterType</td>
</tr>
<tr>
<td>PurchReqOnBehalfReports</td>
</tr>
<tr>
<td>PurchReqOriginationAuthorizationView</td>
</tr>
<tr>
<td>PurchReqProcessingState</td>
</tr>
<tr>
<td>PurchReqQuestionnaireAggregateStatus</td>
</tr>
<tr>
<td>PurchReqQuestionnaireStatus</td>
</tr>
<tr>
<td>PurchReqReportSortOrder</td>
</tr>
<tr>
<td>PurchReqReportStatus</td>
</tr>
<tr>
<td>PurchReqReviewStatus</td>
</tr>
<tr>
<td>PurchReqRFQRequirement</td>
</tr>
<tr>
<td>PurchReqRFQType</td>
</tr>
<tr>
<td>PurchReqSaveChanges</td>
</tr>
<tr>
<td>PurchReqStatus</td>
</tr>
<tr>
<td>PurchReqType</td>
</tr>
<tr>
<td>PurchReqWorkflowState</td>
</tr>
<tr>
<td>PurchRFQQuestionnaireStatus</td>
</tr>
<tr>
<td>PurchRFQStatusVendor</td>
</tr>
<tr>
<td>PurchRFQType</td>
</tr>
<tr>
<td>PurchTableFormId</td>
</tr>
<tr>
<td>PurchTableListPage</td>
</tr>
<tr>
<td>PurchTableMode</td>
</tr>
<tr>
<td>PurchTotalsCachingMethod</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>-----------------------------</td>
</tr>
<tr>
<td>PurchUpdate</td>
</tr>
<tr>
<td>PurchVendorPortalShowResponseType</td>
</tr>
<tr>
<td>QuotationType</td>
</tr>
<tr>
<td>ReqBOMRouteCreated</td>
</tr>
<tr>
<td>ReqCurrentDaySchedFrom</td>
</tr>
<tr>
<td>ReqDemPlanDataSourceType</td>
</tr>
<tr>
<td>ReqDemPlanDemandCategory</td>
</tr>
<tr>
<td>ReqDemPlanForecastAttributeType</td>
</tr>
<tr>
<td>ReqDemPlanForecastingStrategy</td>
</tr>
<tr>
<td>ReqDemPlanForecastType</td>
</tr>
<tr>
<td>ReqDisplayDelay</td>
</tr>
<tr>
<td>ReqForecastReducedBy</td>
</tr>
<tr>
<td>ReqGanttColorType</td>
</tr>
<tr>
<td>ReqGanttShow</td>
</tr>
<tr>
<td>ReqItemJournalType</td>
</tr>
<tr>
<td>ReqItemTableWizardPurpose</td>
</tr>
<tr>
<td>ReqMarkUpdate</td>
</tr>
<tr>
<td>ReqPeggingAssignmentType</td>
</tr>
<tr>
<td>ReqPeggingType</td>
</tr>
<tr>
<td>ReqPlannedOrderLeveling</td>
</tr>
<tr>
<td>ReqPlanType</td>
</tr>
<tr>
<td>ReqPOStatus</td>
</tr>
<tr>
<td>ReqQtyAmount</td>
</tr>
<tr>
<td>ReqRefType</td>
</tr>
<tr>
<td>ReqRefTypeShort</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>-----------------------------</td>
</tr>
<tr>
<td>ReqRefTypeTrunc</td>
</tr>
<tr>
<td>ReqTraceMessageType</td>
</tr>
<tr>
<td>ReqTransFuturesActionPartType</td>
</tr>
<tr>
<td>ReturnCodeType</td>
</tr>
<tr>
<td>ReturnCycleTimeScope</td>
</tr>
<tr>
<td>ReturnReasonCodeDispExtended</td>
</tr>
<tr>
<td>ReturnReasonDispCode</td>
</tr>
<tr>
<td>ReturnUpdateAction</td>
</tr>
<tr>
<td>RouteFormula</td>
</tr>
<tr>
<td>RouteOprPriority</td>
</tr>
<tr>
<td>SalesBasketType</td>
</tr>
<tr>
<td>SalesBatch</td>
</tr>
<tr>
<td>SalesCheckForPickup</td>
</tr>
<tr>
<td>SalesCheckQtyCachKey</td>
</tr>
<tr>
<td>SalesDeliveryTimeState</td>
</tr>
<tr>
<td>SalesDocumentTimezonePreference</td>
</tr>
<tr>
<td>SalesPriceDateType</td>
</tr>
<tr>
<td>SalesPriceModelBasic</td>
</tr>
<tr>
<td>SalesPurchCopy</td>
</tr>
<tr>
<td>SalesPurchCycleAction</td>
</tr>
<tr>
<td>SalesPurchGroup</td>
</tr>
<tr>
<td>SalesPurchParmCleanUpMode</td>
</tr>
<tr>
<td>SalesQuotationFilter</td>
</tr>
<tr>
<td>SalesQuotationLinkToProject</td>
</tr>
<tr>
<td>SalesQuotationListPage</td>
</tr>
</tbody>
</table>
ENUMERATION

SalesQuotationPriceConversion

SalesQuotationPriceSimResult

SalesQuotationTypeListPage

SalesShipping

SalesSourcingOrigin

SalesStatus

SalesTableFormId

SalesTableListPage

SalesTableMode

SalesType

SalesUpdate

ShipCarrierDlvType

ShipCarrierFreightApplied

ShipCarrierMkUpFreight

SMAInvoiceProjectSelection

SMAItemSetupType

SMAProjectSelection

SMAReasonType

SMAServiceBOMChangeAction

SMAServiceFunctionType

SMAServiceLevelAgreementLogType

SMAServiceOrderActionType

SMAServiceOrderFilter

SMAServiceOrderOrigin

SMAServiceOrderProgress
<table>
<thead>
<tr>
<th>ENUMERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMAServiceTaskTitleOption</td>
</tr>
<tr>
<td>SMASubscriptionPeriodType</td>
</tr>
<tr>
<td>SMAWizardCreateType</td>
</tr>
<tr>
<td>smmAccountNumToCreate</td>
</tr>
<tr>
<td>smmActivityParentType</td>
</tr>
<tr>
<td>smmAppointmentNThInstance</td>
</tr>
<tr>
<td>smmBusinessRelationsListFilter</td>
</tr>
<tr>
<td>smmBusRelTypeSourceTable</td>
</tr>
<tr>
<td>smmCampaignBroadcastType</td>
</tr>
<tr>
<td>smmCampaignProjectJournalType</td>
</tr>
<tr>
<td>smmCampaignResponse</td>
</tr>
<tr>
<td>smmCampaignsListFilter</td>
</tr>
<tr>
<td>smmContactsListFilter</td>
</tr>
<tr>
<td>smmCreateOpportunityOptions</td>
</tr>
<tr>
<td>smmDisplayEMailInOutlook</td>
</tr>
<tr>
<td>smmDragDropObjectType</td>
</tr>
<tr>
<td>smmDupMethods</td>
</tr>
<tr>
<td>smmEMailSMS</td>
</tr>
<tr>
<td>smmEntityToDelete</td>
</tr>
<tr>
<td>smmFieldDelimiters</td>
</tr>
<tr>
<td>smmLeadsListFilter</td>
</tr>
<tr>
<td>smmLogType</td>
</tr>
<tr>
<td>smmOpportunitiesListFilter</td>
</tr>
<tr>
<td>smmOpportunityAssociation</td>
</tr>
<tr>
<td>smmOutlookContactDeleteAction</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>---------------------------------</td>
</tr>
<tr>
<td>smmOutlookRecurrenceType</td>
</tr>
<tr>
<td>smmOutlookSyncPrinciple</td>
</tr>
<tr>
<td>smmOutlookUpdateAction</td>
</tr>
<tr>
<td>smmProjectNewExisting</td>
</tr>
<tr>
<td>smmQuotationAccountType</td>
</tr>
<tr>
<td>smmQuotationStatus</td>
</tr>
<tr>
<td>smmRecordDelimiters</td>
</tr>
<tr>
<td>smmSalesUnitMemberRelation</td>
</tr>
<tr>
<td>SmmSourceTypeList</td>
</tr>
<tr>
<td>smmSwotType</td>
</tr>
<tr>
<td>smmTransLogUpdateAction</td>
</tr>
<tr>
<td>smmUpdateOpportunityOptions</td>
</tr>
<tr>
<td>smmWarningError</td>
</tr>
<tr>
<td>SMAActiveAll</td>
</tr>
<tr>
<td>SMAAgreementFilter</td>
</tr>
<tr>
<td>SMAAgreementTableListPageType</td>
</tr>
<tr>
<td>TAMCustRebateApprovalStatus</td>
</tr>
<tr>
<td>TAMFundStatus</td>
</tr>
<tr>
<td>TAMFundType</td>
</tr>
<tr>
<td>TAMPromoCustomerType</td>
</tr>
<tr>
<td>TAMPromoMerchEvent</td>
</tr>
<tr>
<td>TAMPromoMgmtApprovalStatus</td>
</tr>
<tr>
<td>TAMPromotionDate</td>
</tr>
<tr>
<td>TAMPromotionMode</td>
</tr>
<tr>
<td>TAMRebateCustInclusive</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>---------------------------------</td>
</tr>
<tr>
<td>TAMRebateLineBreakType</td>
</tr>
<tr>
<td>TAMRebateStatus</td>
</tr>
<tr>
<td>TAMRebateUnitType</td>
</tr>
<tr>
<td>TAMRebateUOMOption</td>
</tr>
<tr>
<td>TAMVendRebateApprovalStatus</td>
</tr>
<tr>
<td>TAMVendRebateCalcDateType</td>
</tr>
<tr>
<td>TAMVendRebateTakenFrom</td>
</tr>
<tr>
<td>TAMVendRebateTransactionType</td>
</tr>
<tr>
<td>TaxModuleType</td>
</tr>
<tr>
<td>TaxSourceType</td>
</tr>
<tr>
<td>TMSAccessorialAssignmentLevel</td>
</tr>
<tr>
<td>TMSAccessorialAssignmentTarget</td>
</tr>
<tr>
<td>TMSAccessorialDeliveryType</td>
</tr>
<tr>
<td>TMSAccessorialType</td>
</tr>
<tr>
<td>TMSAppointmentAlert</td>
</tr>
<tr>
<td>TMSDiscountResultType</td>
</tr>
<tr>
<td>TMSDiscountType</td>
</tr>
<tr>
<td>TMSFeeType</td>
</tr>
<tr>
<td>TMSFreightBillMatchStatus</td>
</tr>
<tr>
<td>TMSFreightBillReconcileType</td>
</tr>
<tr>
<td>TMSFwkErrorType</td>
</tr>
<tr>
<td>TMSHubPosition</td>
</tr>
<tr>
<td>TMSInvoiceAccountType</td>
</tr>
<tr>
<td>TMSLineType</td>
</tr>
<tr>
<td>TMSLoadBuildSessionState</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>---------------------------</td>
</tr>
<tr>
<td>TMSLoadTender</td>
</tr>
<tr>
<td>TMSLookupType</td>
</tr>
<tr>
<td>TMSNNumberSequenceType</td>
</tr>
<tr>
<td>TMSOverrideLocationType</td>
</tr>
<tr>
<td>TMSRecurrenceDays</td>
</tr>
<tr>
<td>TMSRecurrenceType</td>
</tr>
<tr>
<td>TMSRecurrenceWeeks</td>
</tr>
<tr>
<td>TMSResponsibleForPayment</td>
</tr>
<tr>
<td>TMSRouteStatus</td>
</tr>
<tr>
<td>TMSSalesPurchTransfer</td>
</tr>
<tr>
<td>TMSTableRef</td>
</tr>
<tr>
<td>TMSTransportationType</td>
</tr>
<tr>
<td>TMSTransportRefType</td>
</tr>
<tr>
<td>TMSTransportTypeFilter</td>
</tr>
<tr>
<td>TMSUOM</td>
</tr>
<tr>
<td>TMSZoneType</td>
</tr>
<tr>
<td>TradeCurrencyConversion</td>
</tr>
<tr>
<td>TradeLineDlvType</td>
</tr>
<tr>
<td>TradeNonStockedConversionChangeType</td>
</tr>
<tr>
<td>TradeNonStockedConversionIssue</td>
</tr>
<tr>
<td>TradeNonStockedConversionResolveUndo</td>
</tr>
<tr>
<td>TradePrintType</td>
</tr>
<tr>
<td>TradeTable2LineUpdate</td>
</tr>
<tr>
<td>VendNotificationCategorySelection</td>
</tr>
<tr>
<td>VendNotificationStatus</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>VendPackingSlipTransTimeStatus</td>
</tr>
<tr>
<td>VendRequestCompanyType</td>
</tr>
<tr>
<td>VendRequestOriginatedByType</td>
</tr>
<tr>
<td>VendRequestQuestionnairesCompleted</td>
</tr>
<tr>
<td>VendRequestRoleType</td>
</tr>
<tr>
<td>VendReviewRatingScore</td>
</tr>
<tr>
<td>VersioningAction</td>
</tr>
<tr>
<td>WHSAutoReleaseMaterialOverPick</td>
</tr>
<tr>
<td>WHSAutoReleaseOrderType</td>
</tr>
<tr>
<td>WHSAutoReleaseContainerAtContainerClose</td>
</tr>
<tr>
<td>WHSAutoReleaseContainerAtSmallTime</td>
</tr>
<tr>
<td>WHSAutoReleaseContainerAtLargeTime</td>
</tr>
<tr>
<td>WHSAutoReleaseContainerAtRandom</td>
</tr>
<tr>
<td>WHSContainerizationQueryType</td>
</tr>
<tr>
<td>WHSContainerPackingStrategy</td>
</tr>
<tr>
<td>WHSContainerTableFormViewType</td>
</tr>
<tr>
<td>WHSCrossDockFulfillmentStrategy</td>
</tr>
<tr>
<td>WHSCustJourType</td>
</tr>
<tr>
<td>WHSCycleCountPlanStatus</td>
</tr>
<tr>
<td>WHSDefaultDataField</td>
</tr>
<tr>
<td>WHSFilterModule</td>
</tr>
<tr>
<td>WHSHistoryEvent</td>
</tr>
<tr>
<td>WHSLoadPlanning</td>
</tr>
<tr>
<td>WHSLoadPostMethodsBase</td>
</tr>
<tr>
<td>WhsLoadReplenishment</td>
</tr>
<tr>
<td>WHSLoadTable</td>
</tr>
<tr>
<td>Enumeration</td>
</tr>
<tr>
<td>--------------------------</td>
</tr>
<tr>
<td>WHSMenuItemDirectedBy</td>
</tr>
<tr>
<td>WHSMixedLPReceivingMode</td>
</tr>
<tr>
<td>WHSMobileAppPagePattern</td>
</tr>
<tr>
<td>WHSOriginType</td>
</tr>
<tr>
<td>WHSPickOldestBatch</td>
</tr>
<tr>
<td>WHSPostMethodBaseKanban</td>
</tr>
<tr>
<td>WHSPostMethodBaseKanbanOptional</td>
</tr>
<tr>
<td>WHSPostMethodBaseOptional</td>
</tr>
<tr>
<td>WHSPostMethodBaseProd</td>
</tr>
<tr>
<td>WHSPostMethodBaseProdOptional</td>
</tr>
<tr>
<td>WHSPostMethodsBase</td>
</tr>
<tr>
<td>WHSQtyPct</td>
</tr>
<tr>
<td>WHSReleaseQuantitySpecification</td>
</tr>
<tr>
<td>WHSReplenishmentDependentWorkBlockingPolicy</td>
</tr>
<tr>
<td>WHSReservationHierarchyLevelStrategyType</td>
</tr>
<tr>
<td>WHSReservationStatus</td>
</tr>
<tr>
<td>WHSUseFixedLocations</td>
</tr>
<tr>
<td>WHSWorkActivity</td>
</tr>
<tr>
<td>WHSWorkClusterStatus</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>----------------------------------------</td>
</tr>
<tr>
<td>WHSWorkCreationProcess</td>
</tr>
<tr>
<td>WHSWorkExceptionLogStatus</td>
</tr>
<tr>
<td>WHSWorkExecuteMode</td>
</tr>
<tr>
<td>WHSWorkListPageFilter</td>
</tr>
<tr>
<td>WHSWorkPrintOption</td>
</tr>
<tr>
<td>WHSWorkPutFlow</td>
</tr>
<tr>
<td>WHSWorkTransType</td>
</tr>
<tr>
<td>WHSWorkType</td>
</tr>
<tr>
<td>WHSWorkTypePickPut</td>
</tr>
<tr>
<td>WMSAutoAddStop</td>
</tr>
<tr>
<td>WMSFreightChargeTerms</td>
</tr>
<tr>
<td>WMSFreightCounted</td>
</tr>
<tr>
<td>WMSHandlingType</td>
</tr>
<tr>
<td>WMSLocationType</td>
</tr>
<tr>
<td>WMSPackageType</td>
</tr>
<tr>
<td>WMSPalletMovementProcessing</td>
</tr>
<tr>
<td>WMSPhysicalUpdateStatus</td>
</tr>
<tr>
<td>WMSReceiptStatus</td>
</tr>
<tr>
<td>WMSReservationMethod</td>
</tr>
<tr>
<td>WMSReservationMethodInternal</td>
</tr>
<tr>
<td>WMSShipmentStatus</td>
</tr>
<tr>
<td>WMSShipmentType</td>
</tr>
<tr>
<td>WMSSpaceUtilInconsistencyGroup</td>
</tr>
<tr>
<td>WMSSpaceUtilShowBy</td>
</tr>
<tr>
<td>WMSSpaceUtilStorageLoadUnitType</td>
</tr>
<tr>
<td>ENUMERATION</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>WMSStoreAreaType</td>
</tr>
<tr>
<td>WMSTrailerLoaded</td>
</tr>
<tr>
<td>WrkCtrActivityType</td>
</tr>
<tr>
<td>WrkCtrBulkResReqSearchType</td>
</tr>
<tr>
<td>WrkCtrCapacityType</td>
</tr>
<tr>
<td>WrkCtrCapRefType</td>
</tr>
<tr>
<td>WrkCtrCommitState</td>
</tr>
<tr>
<td>WrkCtrGroupWrkCtr</td>
</tr>
<tr>
<td>WrkCtrSchedulerCommand</td>
</tr>
<tr>
<td>WrkCtrSchedulerConstraintType</td>
</tr>
<tr>
<td>WrkCtrSchedulerLoggerMode</td>
</tr>
<tr>
<td>WrkCtrType</td>
</tr>
<tr>
<td>WrkCtrTypeFilter</td>
</tr>
</tbody>
</table>

These enumerations were removed, and not made extensible.

<table>
<thead>
<tr>
<th>ENUMERATION REMOVED</th>
</tr>
</thead>
<tbody>
<tr>
<td>BackorderLinesListPageMode</td>
</tr>
<tr>
<td>BackorderPurchLinesListPageMode</td>
</tr>
<tr>
<td>EcoResProductPerCompanyListPageType</td>
</tr>
<tr>
<td>ReturnTableListPageType</td>
</tr>
<tr>
<td>SMAAgreementTableListPageType</td>
</tr>
</tbody>
</table>

Foundation changes were made to improve support for extensible enumerations. The `SysPlugin` framework was enabled for enumerations where `IsExtensible` is set to `Yes`. Views were enabled with new name-based syntax for enumerations.

**Data manipulation methods that do not raise DataEvents or missing insert, update, delete pre- and post-data events**

As a general practice, you use data methods on tables to raise events that can be used for extending the application. The code base has not always followed this practice. For example, the `doInsert`, `doUpdate`, and `doDelete` data methods and certain table implementations did not make a call to `super()` in the data method.
The `insert`, `update`, and `delete` methods on the type classes have been refactored. Changes were made so that `super()` is called more consistently in data methods. These changes enable extensions to be added to these methods, so that pre- and post-events are now available for extension. The tables where the `insert`, `update`, and `delete` events were enabled for extension are listed in the following table.

<table>
<thead>
<tr>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>InventBlocking</td>
</tr>
<tr>
<td>InventTransferLine</td>
</tr>
<tr>
<td>Kanban</td>
</tr>
<tr>
<td>KanbanJob</td>
</tr>
<tr>
<td>KanbanJobPickingList</td>
</tr>
<tr>
<td>MCRRoyaltyVendTable</td>
</tr>
<tr>
<td>PdsRebateTable</td>
</tr>
<tr>
<td>PmfProdCoBy</td>
</tr>
<tr>
<td>ProdBOM</td>
</tr>
<tr>
<td>ProdRoute</td>
</tr>
<tr>
<td>ProdTable</td>
</tr>
<tr>
<td>PurchLine</td>
</tr>
<tr>
<td>PurchRFQCaseLine</td>
</tr>
<tr>
<td>PurchRFQCcaseTable</td>
</tr>
<tr>
<td>PurchRFQLine</td>
</tr>
<tr>
<td>PurchTable</td>
</tr>
<tr>
<td>SalesLine</td>
</tr>
<tr>
<td>SalesQuotationLine</td>
</tr>
<tr>
<td>SalesQuotationTable</td>
</tr>
<tr>
<td>SalesTable</td>
</tr>
<tr>
<td>TAMVendRebateTable</td>
</tr>
<tr>
<td>WMSOrder</td>
</tr>
<tr>
<td>WMSOrderTrans</td>
</tr>
</tbody>
</table>
Exposing class members

Additional private members are now available for customization as a result of the changes to the access modifier or new parm methods. The chain of command platform feature enables extension class access to protected methods and members. For more information about chain of command, see *Extensible X++: Chain of Command*.

<table>
<thead>
<tr>
<th>MEMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssetPost.ledgerJournalTrans</td>
</tr>
<tr>
<td>Class DimensionDerivationRule.ledgerDimensionAllocationList</td>
</tr>
<tr>
<td>Class PurchInvoiceJournalCreate.purchTable</td>
</tr>
<tr>
<td>Class PurchTableType.purchTable</td>
</tr>
<tr>
<td>Class PurchTableType.purchTable_orig</td>
</tr>
<tr>
<td>Class SalesInvoiceJournalPost.salesLine</td>
</tr>
<tr>
<td>Class SalesQuotationLineType</td>
</tr>
<tr>
<td>Class SalesQuotationTableType</td>
</tr>
<tr>
<td>Class VendorInvoiceLineSourceDocLineItem.purchLine</td>
</tr>
<tr>
<td>CustCreditLimit.balanceTotalsCalculated</td>
</tr>
<tr>
<td>CustCreditLimit_SalesTable.salesTable</td>
</tr>
<tr>
<td>Form LedgerJournalTransCustPaym.ledgerJournalEngine</td>
</tr>
<tr>
<td>PurchLineType.purchLine</td>
</tr>
<tr>
<td>PurchLineType.purchLine_orig</td>
</tr>
<tr>
<td>SalesLineType.salesLine</td>
</tr>
<tr>
<td>SalesLineType.salesLine_orig</td>
</tr>
<tr>
<td>SalesTableType.checkSalesQty</td>
</tr>
<tr>
<td>SalesTableType.SalesTable_orig</td>
</tr>
<tr>
<td>WHSControl.data</td>
</tr>
<tr>
<td>WHSLocationDirective.targetLicensePlateId</td>
</tr>
</tbody>
</table>

Construct methods with throw statements

Some *construct* methods were implemented with *throw* statements if there was a missing implementation for a given type. This doesn't work well with extensibility, so to mitigate this, *construct* methods were changed so that they do not throw exceptions. These methods are now to open for extensibility through class augmentation or by post-event subscription.
Find methods with throw statements

Some `find` methods were implemented with `throw` statements if there was a missing implementation for a given type. This does not work well with extensibility, so to mitigate this, `find` methods were changed so that they do not throw exceptions. These methods are now to open for extensibility through class augmentation or by post-event subscription.

Methods made hookable

Extensibility support has been extended for some methods that were not public and were not hookable. The following methods have been explicitly decorated with hookable behavior.
**Inline delegates**

Inline delegates are now available. The most common way to use inline delegates is to split the method into more granular methods and enable extensibility events in the smaller methods.
### METHOD

| ForecastSales.ForecastSales_ds.updateForecastSalesFields |
| Form SalesTable - method updateDesign |
| ReqTransPoMarkFirm.firmSelectedPlannedOrders |
| SalesLineType.insert |
| SalesLineType.update |
| SalesTable2LineUpdatePrompt.dialog |
| table ExtCodeTable |
| table InventItemGroup.getGroupForAccountType |
| table InventItemGroup.ledgerDimensionDescription |
| table InventTestAssociationTable |
| Table LedgerJournalName - method validateWrite |
| Table PaymTerm - method due |
| TaxCalculation.newForSourceType |
| TaxCalculation.newForSourceTypeWithTaxUncommitted |
| WHSLoadLine.getOrderCommonFromLoadLine |
| WhsLocationDirective.findLocation |
| WHSRFControlData.populateData |
| WHSRFControlData.processControl |
| WHSRFMenuItemTable.getWHSWorkExecuteMode |

### Other changes

The following table lists additional changes that have been made for extensibility.

<p>| CHANGE |
| Add indirection for existing product dimensions |
| Class FormLetterParmDataOutputContract is not extensible |
| Create an instantiation strategy for the SysExtensionFramework that supports one or more arguments |</p>
<table>
<thead>
<tr>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customization: TableField: Extension Model: Change EDT type of on a table field</td>
</tr>
<tr>
<td>CustVendOpenTransBalances - initAccountNumCurrencies() switch statement</td>
</tr>
<tr>
<td>CustVendOpenTransBalances - new() switch statement</td>
</tr>
<tr>
<td>CustVendOutPaym (Class) needs extensibility improvement</td>
</tr>
<tr>
<td>CustVendPaymReconciliationSetStatus (Class) needs extensibility improvement</td>
</tr>
<tr>
<td>CustVendSumForPaym (Class) needs extensibility improvement</td>
</tr>
<tr>
<td>Decouple AddressCountyId and AddressStateId EDTs from SysGroup</td>
</tr>
<tr>
<td>Document Management event handling needs improved extensibility support</td>
</tr>
<tr>
<td>Exchange rate provider framework requires custom built providers to be placed in the Currency Model</td>
</tr>
<tr>
<td>Extending GS-128</td>
</tr>
<tr>
<td>Extension model: Allow customizations on the CountryRegionCodes property.</td>
</tr>
<tr>
<td>Form extension - Extension of &quot;extended&quot; form elements with new controls are not working</td>
</tr>
<tr>
<td>InventDim: Condition with throw</td>
</tr>
<tr>
<td>InventDimRenameDimValue</td>
</tr>
<tr>
<td>Method overlayering - Class VendInvoiceTableToLineUpdate.convertPurchTableFieldToVendInvoice</td>
</tr>
<tr>
<td>Method overlayering: class Markup - method delete</td>
</tr>
<tr>
<td>Method overlayering: class McrPriceHistoryForm.insertPotentialTradeAgreements</td>
</tr>
<tr>
<td>Method overlayering: Class OffsetVoucher - method markTransaction</td>
</tr>
<tr>
<td>Method overlayering: Form LedgerJournalTransDimension.init</td>
</tr>
<tr>
<td>Method overlayering: Form LedgerTransVoucher - method init</td>
</tr>
<tr>
<td>Method overlayering: Table CustInvoiceTable - method validateWrite</td>
</tr>
<tr>
<td>Method signature changed: RetailCreateLinesFromProductsToAdd.parmCallerCommon</td>
</tr>
<tr>
<td>Method signature changed: WHSInvent.getCommonFromWorkTransType</td>
</tr>
<tr>
<td>Method signature changed: WHSPoolProdBom.movementBuffer</td>
</tr>
<tr>
<td>Missing construct method: class SMAServiceOrderTableButtonStateProvider</td>
</tr>
</tbody>
</table>
Number sequence scope extensibility needed

Runbase needs a way for class extensions to pack/unpack their members

String EDT size extension issues

Support opening Inventory on-hand form based on custom InventDim

SysExtension framework: SysExtensionInstantiationStrategy and SysExtensionAttribute are not compatible

Variations of EventHandlerResult are requested to ensure that delegates used in Request/response scenarios are more robust

WHS Mobile Framework: passes

WhsLocationDirectiveLine To/FromQty not extensible

WHSMobileApp Extensibility

WHSMobileAppAttachedImageDetails.removeLabelFromDimValue() is not generic enough about Product dimensions

WHSMobileAppAttachedImageDetails.removeLabelFromDimValue() is not generic enough about Product dimensions

WhsRFControlData.processControl must reference WhsControl.data instead of _data in the switch block
This topic defines the characteristics of an intrusive customization. Intrusive customizations are the major obstacle to keeping continuous upgrade costs close to zero. Some types of intrusive customizations can be prevented by tooling, whereas other types remain the responsibility of the author of the extension. The X++ compiler and Microsoft Visual Studio designers will prevent some types of intrusive customizations. However, a subset of intrusive customizations can’t be detected by tooling but might still prevent continuous upgrades. Ultimately, the developer is responsible for avoiding intrusive customizations.

A customization that violates any of the following principles is intrusive.

**NOTE**

When extending other solutions you are expected to be responsible. This includes:

- Accepting the responsibility of your extension. You are introducing new behavior and therefore own the full responsibility of the change.
- Allowing other extensions to co-exist. Recognize that you are not the only consumer of an extension point. For example, only respond when needed.
- Only adding extensions that are coherent with the extension point. The longevity of a solution is defined by its resilience to change. Recognize that extension points may be exercised in more or fewer scenarios in future versions. For example, only add relevant validation logic to a validate() method.
- Avoiding dependencies on implementation details. Implementation details are likely to change in future versions. Make your solution resilient by avoiding dependencies on local variables, call-stacks and, calling sequences and avoid using reflection.

**Don't change type definitions**

Types are referenced by their definition. A change to a type's definition is a breaking change and requires that all references be updated. It's impossible to ensure that future references will be implemented correctly (for example, in the model that hosts the type). There are several implications:

- Don't change a method signature. The method signature includes the return type, the name (which includes casing), and the parameters (which include optional parameters).
- Don't change requirements for implementers of interfaces and table maps. For example, don't add a new method to an interface or a new field to a table map.
- Don't change requirements for classes that are derived from abstract classes. For example, don't add a new abstract method to a class.
- Don't reduce access modifiers for types or members. For example, don't change classes, tables, or methods from public to private.
- Don't change constraints that are defined on a table or a data entity. Constraints include allowing editing, mandatory constraints, uniqueness constraints, and referential constraints.

**Don't break encapsulation**

The author of a model must be able improve the product by remaining in control of encapsulated code and types. Model owners must be able to change and delete encapsulated code and types at will, without prior notice, and without risk of downstream impact on extensions and customizations. Encapsulation is broken if, for example, a private method is deleted. Here are some of the implications:
- Don't increase access modifiers for types or members. For example, don't change classes, tables, or methods from private to public.
- If something should be closed for modification, don't open it for customizing behavior. Extension capabilities must be designed as open for extensibility but closed for modification.

**Be additive in nature**

New behaviors are enabled through added functionality as part of extensions. Extension capabilities must be designed in and open for extensions, and they must support multiple extensions that exist side by side in the same installation. There are several implications:

- Don't overlayer. Overlayering replaces the default implementation and prevents multiple solutions from changing the same element.
- Don't significantly change the characteristics of the standard functionality. These characteristics include the user experience and performance. For example, performance must not be adversely affected if an extension is installed but isn't used.
- Don't unconditionally set results in `EventHandlerResult` classes, and don't unconditionally set return values in `XppPrePostArgs` classes.
- Don't replace existing behavior by default, unless the replacing logic conforms to the Liskov Substitution Principle.

**NOTE**

The author of the extension is responsible for not violating these principles.
This article describes the new class extension model in X++.

Because over-layering is a very intrusive feature, we recommend that you not use it. The alternative to over-layering is extension. Extension lets you extend existing artifacts in a new model. Extensions are easier to maintain, but the amount of extension that can be done during customization is limited. There are rich ways to extend the metadata. For example, you can add new fields to a table. This article describes how X++ code can be extended, so that you can add methods and state to artifacts that are defined in other models without recompiling those models. A similar code extension mechanism already exists for X++ and is modeled after the corresponding feature in C#. Under this mechanism, a class can be designated as an extension class through a naming convention and by hosting public static methods. In the existing feature, the type of the first argument that is passed to the extension method is the type to extend. What this article describes is the next step in that direction, which offers a more capable and natural extension story. In object-oriented programming, the term extend has a well-defined meaning. If we say, “class B extends class A,” we mean that B inherits from A, that A is B’s parent class, and the usual object-oriented rules are implied. In fact, this term is even used in the X++ syntax that is used in class declarations to express this relationship. At the same time, we use the term extension to talk about metadata that has contributions from several models. To avoid further overloading the term extend, we will instead use the term class augmentation to designate the relationship between a class A in a base model and a class B in a model that depends on it, where B provides additional functionality to class A in the context of that model. Nevertheless, we will also continue to use the term extension class, because it’s prevalent.

The effective class concept

It’s useful to have a term for a class that consists of the public members of the augmented artifact and all the public members of all the class extensions that augment that artifact. This class is called the effective class in a given model. The following illustration shows an artifact, MyArtifact, that is defined in a base model, MyModel, and two dependent models that have extension classes for MyArtifact.
In this example, the effective class is the class in the extension models that contains all the original methods and all the public artifacts from the extension classes. The effective class isn't the same in every model because it includes only the class extensions that are defined in a given model. The following illustration shows the effective class of `MyArtifact` in the `MyExtensionModel` model.

We will describe class extensions by using a class that is named `MyClass` in a model that is named `MyModel`.

```java
class MyClass {
    public int mycState;
    public str mycMethod(int _arg) {
        // ...
    }
}
```

We can add new methods and state to `MyClass` by introducing an extension class in the extension model (`MyExtensionModel`) that builds on top of (that is, has a dependency on) `MyModel`. 
Extension class declarations

Extension classes are final classes that are adorned with the `ExtensionOf` attribute and that also have a name that has the `_Extension` suffix. (This restriction on the naming might be removed later.) The name of the extension class is otherwise unimportant. The class augments the artifact that is specified in the parameter of the `ExtensionOf` attribute, as shown in the following example.

```csharp
[ExtensionOf(classStr(MyClass))]
final class MyClass_Extension
{
    private void new()
    {
    }
}
```

Because the classes are instantiated by the runtime system, it's not meaningful to derive from the extension class. Therefore, the extension class must be marked as `final`. The extension class `MyClass_Extension` does not extend the designated class (`MyClass`). Therefore, you cannot override methods from `MyClass` in `MyClass_Extension`. The `classStr` compile-time function must be used to designate the augmented class, and it serves two purposes:

- It produces a compilation error if the `MyClass` class doesn't exist.
- The compile-time function that is used tells the compiler what kind of artifact is augmented. Artifact names by themselves don't uniquely identify a given artifact to augment. For example, forms can have the same names as tables, classes, and enums.

Any number of extension classes can augment a given artifact in a particular model. Extension classes are never referenced directly by the programmer, only by the runtime system.

Extension class inheritance

Any class that inherits from an augmented class also inherits the effective class. In other words, the classes that inherit from a class that has extensions inherit the methods that are defined in the extension classes.

Constructors

X++ supports both instance constructors and static constructors.

**Instance constructors**

The instance constructor is the method that is named `new`. Constructors are useful for initializing the state of the extension objects. The instance constructor that is defined in an extension class can't have parameters. Instances of the extension classes are created, and the runtime system calls their constructors as required by the usage scenario. These constructors are never explicitly called by your code. It's guaranteed that the constructor that is provided in an extension class will be called once before any instance method or the instance state on the extension class is accessed. However, if no such references are made, the constructor isn't called.

**Static constructors**

Static constructors are the parameter-less static methods that are named `typenew`. Static constructors can be defined on extension classes. It's guaranteed that the runtime system will call the constructor exactly once before the first reference to the extension type. You can't assume any particular order of invocation for static construction among a set of extensions. This means that you should be careful about referencing static data from other classes in static constructors.

Methods
The public methods that are defined in extension classes provide additional functionality to the augmented class in the context of the model where the extension class is defined. Only public methods are exposed in this way. You can define private methods to help implement the public methods, but those private methods aren't part of the effective class. Because extension classes are final, methods should not be marked as protected.

**Instance methods**

The following example defines an extension method named `ExtensionMethod` in a class that augments `MyClass`.

```java
[ExtensionOf(classStr(MyClass))]
final class MyClass_Extension
{
    private void new()
    {
    }
    public int ExtensionMethod(int arg)
    {
    }
}
```

The public instance method (`ExtensionMethod`) is defined in the extension class. Therefore, it's available just as if it were defined in `MyClass` in the context of the model where the extension class is defined. The following example shows how to call the method in the model.

```java
MyClass c = new MyClass();
print c.ExtensionMethod(32);  
```

Note that the instance method that is defined in the extension class is used as an instance method on the augmented artifact. An extension method can access public and protected members only from the artifact that it augments. This behavior is by design. No artifact should be able to interact directly with state and methods that are explicitly hidden through the `private`, or `internal` keywords. Otherwise, direct interaction with explicitly hidden state and methods could cause malfunction by invalidating key implementation assumptions in those artifacts. Methods and statements in the method body can use the `this` keyword. In this context, the type of `this` is the effective class of the augmented artifact.

**Static methods**

Methods that are defined as public and static in the extension class are available as static methods on the artifact that is augmented.

```java
[ExtensionOf(classStr(MyClass))]
final class MyClass_Extension
{
    private void new()
    {
    }
    public int method1(int arg)
    {
    }
    public static real CelsiusToFahrenheit(real celsius)
    {
        return (celsius * 9.0 / 5.0) + 32.0;
    }
}
```

The following example shows how to call the method in the model.
A static method can access the public static methods and state in the effective class of the augmented artifact. As an interesting side effect, static extension methods on the `Global` class become available in the language as functions, which are available without any prefix.

## State

In addition to providing static and instance methods to an artifact, you can add instance state and static state.

### Instance state

Instance state, which is state that pertains to a particular instance of an artifact, can be specified on extension classes. The following example defines a state that is named `state`.

```csharp
[ExtensionOf(classStr(MyClass))]
final class MyClass_Extension
{
    public int state;
    private void new()
    {
    }
}
```

The following example shows how to use `state` in your code.

```csharp
MyClass c = new MyClass();
c.state = 12;
```

### Static state

Static state applies to the type, not instances of the type. The following example defines a static member called `staticState` in the Extension class augmenting the `MyClass` class.

```csharp
[ExtensionOf(classStr(MyClass))]
final class MyClass_Extension
{
    public int state;
    public static int staticState;
    static void TypeNew()
    {
        staticState = 77;
    }
}
```
The functionality for class extension, or class augmentation, has been improved. You can now wrap logic around methods that are defined in the base class that you're augmenting. You can extend the logic of public and protected methods without having to use event handlers. When you wrap a method, you can also access public and protected methods, and variables of the base class. In this way, you can start transactions and easily manage state variables that are associated with your class.

For example, a model contains the following code.

```java
class BusinessLogic1 {
    str doSomething(int arg) {
        // ...
    }
}
```

You can now augment the functionality of the `doSomething` method inside an extension class by reusing the same method name. An extension class must belong to a package that references the model where the augmented class is defined.

```java
[ExtensionOf(classStr(BusinessLogic1))]
final class BusinessLogic1_Extension {
    str doSomething(int arg) {
        // Part 1
        var s = next doSomething(arg + 4);  
        // Part 2
        return s;
    }
}
```

In this example, the wrapper around `doSomething` and the required use of the `next` keyword create a Chain of Command (CoC) for the method. CoC is a design pattern where a request is handled by a series of receivers. The pattern supports loose coupling of the sender and the receivers.

We now run the following code.

```java
BusinessLogic1 object = new BusinessLogic1();
info(object.doSomething(33));
```

When this code is run, the system finds any method that wraps the `doSomething` method. The system randomly runs one of these methods, such as the `doSomething` method of the `BusinessLogic1_Extension` class. When the call to the next `doSomething` method occurs, the system randomly picks another method in the CoC. If no more wrapped methods exist, the system calls the original implementation.
Capabilities

The following sections give more details about the capabilities of method wrapping and CoC.

Wrapping public and protected methods

Protected or public methods of classes, tables, data entities, or forms can be wrapped by using an extension class. The wrapper method must have the same signature as the base method.

- When you augment form classes, only root-level methods can be wrapped. You can’t wrap methods that are defined in nested classes.
- Currently, only methods that are defined in regular classes can be wrapped. Methods that are defined in extension classes can’t be wrapped by augmenting the extension classes. This capability is planned for a future update.

What about default parameters?

Methods that have default parameters can be wrapped by extension classes. However, the method signature in the wrapper method must not include the default value of the parameter.

For example, the following simple class has a method that has a default parameter.

```java
class Person
{
    public void salute(str message = "Hi") {...}
}
```

In this case, the wrapper method must resemble the following example.

```java
[ExtensionOf(classStr(Person))]
final class APerson_Extension
{
    public void salute(str message)
    {
        // ...
    }
}
```

In the APerson_Extension extension class, notice that the salute method doesn’t include the default value of the message parameter.

Wrapping instance and static methods

Instance and static methods can be wrapped by extension classes. If a static method is the target that will be wrapped, the method in the extension must be qualified by using the static keyword.

For example, we have the following A class.
class A
{
    public static void aStaticMethod(int parameter1)
    {
        // ...
    }
}

[ExtensionOf(classStr(A))]
final class An_Extension
{
    public static void aStaticMethod(int parameter1)
    {
        // ...
        next aStaticMethod(parameter1);
    }
}

**IMPORTANT**
The ability to wrap static methods doesn't apply to forms. In X++, a form class isn't a new class, and can't be instantiated or referenced as a normal class. Static methods in forms don't have any semantics.

**Wrapper methods must always call next**
Wrapper methods in an extension class must always call `next`, so that the next method in the chain and, finally, the original implementation are always called. This restriction helps guarantee that every method in the chain contributes to the result.

In the current implementation of this restriction, the call to `next` must be in the first-level statements in the method body.

Here are some important rules:

- Calls to `next` can't be done conditionally inside an `if` statement.
- Calls to `next` can't be done in `while`, `do-while`, or `for` loop statements.
- A `next` statement can't be preceded by a `return` statement.
- Because logical expressions are optimized, calls to `next` can't occur in logical expressions. At runtime, the execution of the complete expression isn't guaranteed.

**NOTE**
If a method is replaceable, extenders don't have to unconditionally call `next` when wrapping the method by using chain of command. Although extenders can break the chain, the expectation is that they will only conditionally break it. The compiler doesn't enforce calls to `next` for methods with the attribute, `Replaceable`.

**Wrapping a base method in an extension of a derived class**
The following example shows how to wrap a base method in an extension of a derived class. For this example, the following class hierarchy is used.
class A
{
    public void salute(str message)
    {
        info(message);
    }
}
class B extends A {}
class C extends A {}

Therefore, there is one base class, A. Two classes, B and C, are derived from A. We will augment or create an extension class of one of the derived classes (in this case, B), as shown here.

[ExtensionOf(classStr(B))]
final class B_Extension
{
    public void salute(str message)
    {
        next salute(message);
        info("B extension");
    }
}

Although the B_Extension class is an extension of B, and B doesn’t have a method definition for the salute method, you can wrap the salute method that is defined in the base class, A. Therefore, only instances of the B class will include the wrapping of the salute method. Instances of the A and C classes will never call the wrapper method that is defined in the extension of the B class.

This behavior becomes clearer if we implement a method that uses these three classes.

class ProgramTest
{
    public static void main(Args args)
    {
        var a = new A();
        var b = new B();
        var c = new C();

        a.salute("Hi");
        b.salute("Hi");
        c.salute("Hi");
    }
}

For calls to a.salute("Hi") and c.salute("Hi"), the Infolog shows only the message “Hi.” However, when b.salute("Hi") is called, the Infolog shows “Hi” followed by “B extension.”

By using this mechanism, you can wrap the original method only for specific derived classes.

Accessing protected members from extension classes

As of Platform update 9, you can access protected members from extension classes. These protected members include fields and methods. Note that this support isn’t specific to wrapping methods but applies all the methods in the class extension. Therefore, class extensions are more powerful than they were before.

The Hookable attribute

If a method is explicitly marked as [Hookable(false)], the method can’t be wrapped in an extension class. In the following example, anyMethod can’t be wrapped in a class that augments AnyClass1.
NOTE
For compatibility reasons, `[Hookable(false)]` overrides the behavior of chain of command in addition to pre- and post-handlers. However, `[Hookable(true)]` only applies to pre- and post-handlers and does not influence chain of command wrapping.

Final methods and the Wrappable attribute
Public and protected methods that are marked as `final` can't be wrapped in extension classes. You can override this restriction by using the `Wrappable` attribute and setting the attribute parameter to `true` ([`Wrappable(true)`]). Similarly, to override the default capability for (non-final) public or protected methods, you can mark those methods as non-wrappable ([`Wrappable(false)`]).

In the following example, the `doSomething` method is explicitly marked as non-wrappable, even though it's a public method. The `doSomethingElse` method is explicitly marked as wrappable, even though it's a final method.

```java
class AnyClass2
{
    [Wrappable(false)]
    public void doSomething(str message) {...}

    [Wrappable(true)]
    final public void doSomethingElse(str message) {...}
}
```

Extensions of form-nested concepts such as data sources, data fields, and controls
In order to implement CoC methods for form-nested concepts, such as data sources, data fields, and controls, an extension class is required for each nested concept.

Form data sources
In this example, `FormToExtend` is the form, `DataSource1` is a valid existing data source in the form, and `init` and `validateWrite` are methods that can be wrapped in the data source.
In this example, a data field is extended. **FormToExtend** is the form, **DataSource1** is a data source in the form, **Field1** is a field in the data source, and **validate** is one of many methods that can be wrapped in this nested concept.

In this example, **FormToExtend** is the form, **Button1** is the button control in the form, and **clicked** is a method that can be wrapped on the button control.

Requirements and considerations when you write CoC methods on extensions for form-nested concepts

- Like other CoC methods, these methods must always call `next` to invoke the next method in the chain, so that the chain can go all the way to the kernel or native implementation in the runtime behavior. The call to `next` is equivalent to a call to `super()` from the form itself to help guarantee that the base behavior in the runtime is always run as expected.

- Currently, the X++ editor in Microsoft Visual Studio doesn’t support discovery of methods that can be
Extensions of tables and data entities

An extension class is required for each concept.

Tables

In this example, `TableToExtend` is the table and `delete`, `canSubmitToWorkflow`, and `caption` are methods that can be wrapped in the table.

```
[ExtensionOf(tablestr(TableToExtend))]  
final class TableToExtend_Extension  
{
    public void delete()  
    {
        next delete();  
        //...  
    }  

    public boolean canSubmitToWorkflow(str _workflowType)  
    {
        boolean ret;  
        //...  
        ret = next canSubmitToWorkflow(_workflowType);  
        //...  
        return ret;  
    }  

    public str caption()  
    {
        str ret;  
        //...  
        ret = next caption();  
        //...  
        return ret;  
    }  
}
```

Data entities

You cannot add CoC to wrap methods that aren’t defined in the original base behavior of the nested control type. For example, you can’t add `methodInButton1` CoC on an extension. However, from the control extension, you can make a call into this method if the method has been defined as public or protected. Here is an example where the `Button1` control is defined in the `FormToExtend` form in such a way that it has the `methodInButton1` method.

```
public class FormToExtend extends FormRun  
{
    [Control("Button")]  
    class Button1  
    {
        public void methodInButton1(str param1)  
        {
            info("Hi from methodInButton1");  
            //...  
        }  
    }  
}
```

You do not have to recompile the module where the original form is defined to support CoC methods on nested concepts on that form from an extension. For example, if the `FormToExtend` form from the previous examples is in the `ApplicationSuite` module, you don’t have to recompile `ApplicationSuite` to extend it with CoC for nested concepts on that form from a different module.

Extensions of tables and data entities

An extension class is required for each concept.

Tables

In this example, `TableToExtend` is the table and `delete`, `canSubmitToWorkflow`, and `caption` are methods that can be wrapped in the table.
In this example, **DataEntityToExtend** is the data entity and **validateDelete** and **validateWrite** are methods that can be wrapped in the data entity.

```java
[ExtensionOf(tableStr(DataEntityToExtend))]
final class DataEntityToExtend_Extension
{
    public boolean validateDelete()
    {
        boolean ret;
        //...
        ret = next validateDelete();
        //...
        return ret;
    }

    public boolean validateWrite()
    {
        boolean ret;
        //...
        ret = next validateWrite();
        //...
        return ret;
    }
}
```

**Restrictions on wrapper methods**

The following sections describe restrictions on the use of CoC and method wrapping.

**X++ classes that are compiled by using Platform update 8 or earlier**

The method wrapping feature requires specific functionality that is emitted by an X++ compiler that is part of Platform update 9 or later. Methods that are compiled by using earlier versions don’t have the infrastructure to support this feature.

**Methods on types nested within forms can be wrapped in Platform update 16 and later**

The ability to wrap methods on types nested within forms (data sources and controls) by using class extensions was added in Platform update 16. This means that Chain of Command can be used to provide overrides for data source methods and form control methods.

However, wrapping (extension) of purely X++ methods on those nested types (form controls and form data sources) is not yet supported like it is on other types (forms, tables, data entities). Currently, if a developer uses Chain of Command on purely X++ methods on types inside forms, then it compiles, but the extension methods are not invoked at runtime. This capability is planned for a future update.

**Unimplemented system methods on tables and data entities can be wrapped in Platform update 22 and later**

The ability to wrap methods in nested classes by using class extensions was added in Platform update 16. The concept of nested classes in X++ applies to forms for overriding data source methods and form control methods.

**Next calls can be put inside try/catch/finally in Platform update 21 and later**

In a CoC extension method, the next call must not be called conditionally. However, in Platform update 21 and later next calls can be placed inside a try/catch/finally to allow for standard handling of exceptions and resource cleanup.
public void someMethod()
{
    try
    {
        //...
        next updateBalances();
        //...
    }
    catch(Exception::Error)
    {
        //...
    }
}

Extensions of extensions are not yet supported
Currently, only methods that are defined in regular classes can be wrapped. Methods that are defined in extension classes can't be wrapped by augmenting the extension classes. This capability is planned for a future release.

Extensions of constructors
Constructors cannot be extended. A new method that is defined on an extension class will define a constructor for the extension class itself. Additionally, the new method has to be public, and it can't have any arguments. For more information, see Constructors.

Tooling
For the features that are described in this topic, the Microsoft Visual Studio X++ editor doesn't yet offer complete support for cross-references and Microsoft IntelliSense.
High level guidance: use prefixes to reduce conflicts and improve identification

Naming model elements

Every element in a model must have a name that is unique across all models at installation time. However, at installation time, you don’t know the names of all the models that your model might be installed together with. To accommodate this situation, every element name should include a prefix that is specific to your solution. By including this prefix when you name elements in your model, you significantly reduce the risk of naming conflicts.

- If a model contains multiple solutions, each solution in the model can be identified by a different prefix.
- You must carefully choose the prefix to minimize the risk that other models from other parties use the same prefix for their elements.

When you extend functionality in other models, elements that are being extended already contain a prefix. However, you should not add your prefix to the extension elements, so that the names include multiple successive prefixes. Instead, you should include your prefix or another term or abbreviation as an infix when you name extension elements.

Naming extensions

An extension element, such as a table extension, view extension, or form extension, must have a unique name that minimizes the risk of conflicts with extensions in other models. To minimize the risk of conflicts, the name should include a term, abbreviation, or infix that distinguishes the extension from other extensions to the same element in other models.

- Include either the name of the model where the extension element resides or the prefix that the extension is associated with. For example, a Warehousing module extends the HCMWorker table and uses the WHS prefix in the name of all other elements. In this case, the extension might be named HCMWorker.WHSExtension. Notice that the prefix that is used to name other elements in the module is inserted as an infix in the name. As another example, an extension of the ContactPerson table in the ContosoCustomizations model might be named ContactPerson.ContosoCustomizations if the extension is intended to contain all extensions to the ContactPerson table from the ContosoCustomizations model. The developer tools will default to using the model's name as the extension name, since the model name is already required to be unique.
- Don’t name the extension just <Element that is being extended>.Extension. For example, an extension of the InventLocation table must not be named InventLocation.Extension, because the risk of conflicts is too high.

Naming extension classes

Extension classes that are used to augment the logic on tables, classes, or other elements must have a name that is unique across all types in all models. Preferably, the extension class should include the name of the type that is being extended. However, the name must also include a term, abbreviation, or prefix that distinguishes the class from other types.

- Start the name of the extension class with the name of the type that is being augmented, and end the name...
Naming fields, field groups, indexes, relations, and metadata elements added in extensions

Fields, field groups, indexes, relations, and metadata elements added in extensions must have a name that is unique across both the element that is being extended and other extension elements. Therefore, these artifacts should include a prefix that minimizes the risk of conflicts across models. In addition, these artifacts should have clear terms and abbreviations so that they can be easily understood.

- Include a prefix, term, or abbreviation at the beginning of the name of the metadata node. For example, an approving worker foreign key field is added as part of a table extension, and WHS is one of prefixes that are dedicated to other elements in the hosting model. In this case, the field might be named WHSApprovingWorker.

Naming variables and methods added in extension classes

Variables and methods added in extension classes must have a name that is unique across both the type that is being extended and all other extension classes that extend the same type. You should take care to create unique and readable names for variables and methods. When you can’t create a unique name, then you should apply a prefix to minimize the risk of conflicts across models.

- Create unique and readable names for variables and methods. For example, an approving worker class-level member variable might be named approvingWorkerForLocalWarehouse if the area of functionality is related to supporting some local warehouse extension functionality. An approveWork method for the same area of functionality might be named approveWorkForLocalWarehouse.

- When you cannot create a unique and readable name, then add a prefix, term, or an abbreviation at the beginning of the member variable or method name. For example, an approving worker class-level member variable might be named whsApprovingWorker if WHS is one of the prefixes that is used by other elements in the model. An approveWork method might be named whsApproveWork if WHS is a prefix that is used by other elements in the hosting model.

- Avoid generic names, because the risk is high that multiple extensions could be using the same term or that the base functionality would be enhanced with an identical name in a future release. Some example of names likely to collide are Approver, Delay, Group, Lookup, and Process.
Development tools for Finance and Operations apps, starting with version 8.0, do not allow Microsoft code to be customized by using over-layering. Instead, extension capabilities should be used to modify and add behavior. The "no over-layering" restriction is a key part of the evolution of the product toward providing customers with a cloud service that is simple to update and always running the most recent version possible to allow all customers to receive the benefits of the latest features and fixes.

After you upgrade code from Dynamics AX 2012 or from Dynamics 365 for Finance and Operations version 7, any customizations that still use over-layering will cause errors when you compile your code. To refactor the code, the over-layering restriction can be temporarily relaxed in the model descriptor file of the model that is being over-layered. This temporary relaxation only works on development and demo environments and cannot be deployed on runtime environments like Standard Acceptance Test (or higher) sandbox or production environments. Relaxing the descriptor restriction will enable the code to be gradually refactored to extensions, compiled, run, and then tested.

**Detailed process**

Complete the following steps to relax model restrictions. This procedure can be completed on a cloud environment or a local virtual machine (VM).

1. Deploy the development environment for the Finance and Operations app.

2. Run the Lifecycle Services (LCS) code upgrade service to upgrade the solution.

3. Temporarily allow over-layering in Microsoft models as needed to enable compilation.
   a. Locate the desired model within the C:\AOSService\PackagesLocalDirectory folder.
   b. Navigate to the descriptor folder. For example, \Currency\Descriptor.
   c. Open the XML file. For example, Currency.xml.
   d. Add a `Customization` metadata element inside the `AxModelInfo` metadata element to indicate `AllowAndWarn` so that the start of the file now looks like this.

   ```xml
   <AxModelInfo xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
   <Customization>AllowAndWarn</Customization>
   </AxModelInfo>
   ```

4. Refactor over-layering to extensions and test. Make use of extension capabilities to eliminate over-layering. If needed, make extensibility requests.

5. Revert the temporary changes to Microsoft models. The deployment of a model that uses over-layering will not be possible, so it's important to ensure that the descriptor file is updated.

**Prototyping extensibility requests**

As a solution gradually migrates toward extensions, there will be places where an extensibility request is required to unblock the solution. To fully understand what is needed to unblock a solution and smooth the
migration process toward extensions, extension capabilities can be prototyped in a separate model.

1. Identify the need for an extension capability to refactor some over-layering.

2. Add a prototype of extension capabilities into an extension prototype model.
   - For enumeration extensibility, put a copy of the enumeration into the extension prototype model and mark it as extensible.
   - For extract method extensibility, prototype the extracted method and the overlayered original in the extension prototype model.

3. Use the prototype extension capability.

4. If the prototype extension capability is sufficient, create a matching request.

5. When the extension capability has been implemented in the platform or application, the prototype extension capability and any associated over-layering should be removed from the extension prototype model.

6. After the solution has all over-layering refactored to extensions and the extension prototype model is empty, the solution refactoring is complete.
Add values to enums through extension

To add new values to an enum, you should extend the enum. Any enum that is marked as Extensible (IsExtensible = true) can be extended. You can find the extensibility information in the Properties window in Microsoft Visual Studio, as shown in the following illustration.

![Properties window in Visual Studio](image)

When enum values of extensible enums are synchronized, the integer values of the baseline enum are deterministic, whereas the integer values of the extension enum values are non-deterministic. The values are generated during synchronization. Therefore, you can't have logic that depends on the integer value of the enum values. Here are some examples:

- Range comparisons, such as <, >, and ..
- Modeled ranges in views and queries
- Query ranges that are created from code

Usually, an extended enum must have its own implementation wherever it’s used. Look for all uses of the enum, and uptake the implementation where it’s required. Here are some significant places to look for:

- Switch blocks:
  - If the switch block doesn’t have a default case block or a default case block that doesn’t throw an exception, handle the extended enum value by subscribing to a delegate, if a delegate is provided. Otherwise, add a post-event handler to the method.
  - If the enum is used in a switch that has a default case block that throws an exception, contact Microsoft to request a delegate.
- If the enum has an associated class hierarchy that handles the enum, create a subclass for the extended enum-specific implementation, and uptake the construct on the base class as required. For more information, see Register subclasses for factory methods.

Extend an enum

There are two ways to extend an enum:

- Create a project that has a model reference where you want the new enum extension. Right-click the enum to extend, and then select **Create extension**.
Right-click the enum to extend, and then select **Create extension in new project**. You're prompted to select the model that the extension enum should be created in.

The enum extension is created in the selected model. You can add new enum values to this extension.
There are several properties that can be customized on existing extended data types (EDTs) through extension:

- Label
- Help text
- Form help
- Country region codes
- String size
  - You can only modify the value if the EDT does not extend from another EDT.
  - You can only set the new String size to a value equal to or larger than the base EDT value.
- Decimals (NoOfDecimals property)
  - For more information, see Extending decimal point precision for selected data types.

You modify the properties as you would for newly added elements, using the property sheet.

After compiling the code, you can see the changes in the application.

You can view the created extensions in the Application Explorer in Visual Studio.
If the EDT is modified in more than one model

If multiple ISVs have extended the same extended data type, the properties of the EDT from the model with the highest Model ID (closest to USR) will be used. If there are multiple models with changes in the same layer, changes from the model with the highest Model ID will be used. For example, if ISV 1 modified the label of ItemId to “Awesome item number” in model AwesomeModel (USR layer) with ID 15, while ISV 2 modified the label of ItemId to “Super item number” in model SuperModel (USR layer) with ID 12, the end user would see “Awesome item number” in the user interface instead of “Item number”.

NOTE

Instead of extending an existing EDT, you can create a new one, deriving it from the existing EDT. This allows you to edit more properties than you could edit using the extension approach. This means that you would need to modify the fields using this EDT to use your new EDT.
Class inheritance is a central concept in X++, as in other object-oriented languages. The object-oriented strategy pattern is used throughout the X++ business logic. In this pattern, variations in behavior can be encapsulated by subclasses, and the business process uses an abstract base class or interface. A factory method determines the variation that is used, by creating an instance of a specific subclass.

This topic describes how to register your own variations for the factories.

In X++, the factories use reflection to perform the following tasks:

- Find the correct subclass. The factory uses an extension framework to search all subclasses in a hierarchy for a specific set of attributes. If the attributes that decorate a subclass match the parameters that were passed to the factory, that specific class is used.
- Create an instance. After the right type is identified, reflection is used to create an instance of the class.

The following illustrations shows a typical decorated hierarchy.

In X++, two extension frameworks serve the same purpose. The implementer of the factory method determines which extension framework should be used:

- **SysExtension**
  - This extension framework uses custom attributes that make it easier to consume.
  - It supports singletons that save a little performance when the same instance is created repeatedly. This extension framework is especially useful for stateless subclasses.
  - It seamlessly supports extensible enums that are often used to determine the variant.

- **SysPlugin**
  - This extension framework is based on the Managed Extension Framework. The Managed Extension Framework makes the SysPlugin extension framework available to non-X++ code.
  - It uses the `ExportMetadataAttribute` attribute to decorate the variants and is string-based.
  - It uses the `ExportAttribute` attribute to make the variant discoverable.
Introduce a new variant

1. Identify the base class (or interface) of the variant that you must implement.

2. Create a new subclass of the base class, and implement your variation.

3. Identify which attribute is required in order to register your class. There are two approaches:
   
   - Look for attributes that are defined on other subclasses in the hierarchy.
   - Look at the implementation of the factory method. That implementation will contain the attributes that the factory method is searching for.

4. Decorate your subclass with the attribute that was used to match your variation.

SysExtension example

```csharp
[SysWorkExecuteMode(SysWorkExecuteMode::About)]
class WHSWorkExecuteDisplayAbout extends WHSWorkExecuteDisplay
{
   // Your code here.
}
```

SysPlugin example

```csharp
[ExportMetadataAttribute('CaseIAssociation', 'Lead'), ExportAttribute('Dynamics.AX.Application.CaseIAssociation')]
class smmLeadCaseAssociationProvider implements CaseIAssociation
{
   // Your code here.
}
```
Some delegate methods are implemented so that they can request a response from subscribing delegate handler methods. The delegate calling logic then uses the response from a potential subscriber when it continues execution after the response has been received. These delegate methods usually have a signature that has an `EventHandlerResult` parameter as the last parameter. However, because of the support for the `EventHandlerAcceptResult` and `EventHandlerRejectResult` types, the parameter can be of any type that implements the `IEventHandlerResult` interface.

- In general, the logic that is implemented in the delegate handler method should contain a condition that verifies that the subscribing logic is responsible for providing a response. It should also include logic to provide the response in the form of a result.
- When the delegate handler method must provide the response to an `EventHandlerResult` object parameter, the subscribing logic might also contain logic to calculate or retrieve the result.
- When the condition and the response logic are implemented, the calculation of the result must occur only when the condition is evaluated to `true`.
- All the subscribing delegate handler methods are run when a delegate is called. Therefore, you should make sure that the overhead of running your method is as low as possible when the method isn't responsible for providing a response. Therefore, make sure that the condition is evaluated to `false` as quickly as possible when your delegate handler method isn't responsible for providing a result.

**Examples**

The following example shows a delegate handler that has a condition in the form of a `switch` statement. The delegate handler also has logic to provide a response in the form of the result. The responding logic is run only when the condition is evaluated to `true`.

```csharp
public static void validateWarehouseTypeIsSupportedStandardDelegateHandler(InventLocationType _inventLocationType, EventHandlerResult _result)
{
    switch (_inventLocationType)
    {
        case InventLocationType::Standard:
        case InventLocationType::Quarantine:
        case InventLocationType::Transit:
            _result.result(true);
            break;
        }
}
```

When the delegate method requests a response by using an `EventHandlerAcceptResult` or an `EventHandlerRejectResult` object parameter, the subscriber is expected to respond only with an accept or a reject. The subscribing logic might also add messages to the Infolog.

The following example resembles the previous example. However, the delegate method now requests a response by using an `EventHandlerAcceptResult` object and by calling the `accept` method.
public static void validateWarehouseTypeIsSupportedStandardDelegateHandler(InventLocationType _inventLocationType, EventHandlerAcceptResult _result) {
    switch (_inventLocationType) {
    case InventLocationType::Standard:
    case InventLocationType::Quarantine:
    case InventLocationType::Transit:
        _result.accept();
        break;
    }
}

The following example shows a delegate handler method that responds by using an EventHandlerRejectResult object. To respond by using an EventHandlerRejectResult object, you can call the reject method or the checkFailed extension method. If you use the checkFailed method, you can add a warning message to the Infolog. Internally, the checkFailed method calls the reject method.

Guidelines

In addition to the previously described practices, the following general guidelines apply:

- Respond only when the subscribing logic is responsible for responding. The delegate handler methods were implemented to provide a response when a specific condition is met. Therefore, the subscribing logic must provide a result when a specific condition is met. Before the subscribing logic responds, it should not evaluate whether the result object parameter already contains a result. For example, a delegate handler method should not contain logic that resembles the logic in the following example. This logic evaluates whether the EventHandlerResult object parameter already contains a result when the method is run.

WARNING
This example is an example of code that you should not write.
public static void validateWarehouseTypeIsSupportedStandardDelegateHandler(InventLocationType _inventLocationType, EventHandlerResult _result)
{
    switch (_inventLocationType)
    {
    case InventLocationType::Standard:
        case InventLocationType::Quarantine:
        case InventLocationType::Transit:
            _result.result(true);
            break;
    // this default block is an example of the bad practice
    default:
        _result.result(false);
        break;
    }
}

• Don’t provide a response on behalf of other subscribers. If the delegate handler method isn’t responsible for providing a response, the method must not provide a response. If the method provides a response when the condition isn’t met, it provides a response on behalf of other subscribers. The requesting logic must be responsible for handling situations where no subscribers have responded. The delegate handler method must not contain logic that resembles the logic in the following example. This logic which provides a result when the condition is evaluated to false.

**WARNING**

This example is an example of code that you should not write.
When you extend functionality of the application suite, you will encounter classes that extend the RunBase class. This topic shows how a RunBase class can be augmented end to end.

For example, you want to extend the SysUserLogCleanup class. Out of the box, this class can delete records from the SysUserLog table. However, you want to archive these records to a different table before they are deleted.

The SysUserLogCleanup class is a RunBase class. The RunBase class has a dialog box, where the user is prompted for parameters before the class is run. For this example, we will add a toggle button control to the dialog box, get the value of the control, act on the value in the run method, and make sure that the value is serialized via the pack and unpack methods. Serialization helps guarantee that the user’s last selection is presented again if the dialog box is reopened. It also helps guarantee that the settings are applied if the class is run in the background.

To avoid collisions with other eventual extensions, we followed these best practices:

- **Prefix members and methods.** In the example, the prefix “my” is used. This practice is important, because it helps prevent name clashes with other extensions and future versions of the augmented class.

- **Use RunBase.packExtension() and RunBase.unpackExtension().** These methods provide serialization in a nonintrusive manner. They enable serialization of multiple extensions of the same class. The methods are available starting in Platform Update 5.

The following example shows how to implement this scenario.

```java
[ExtensionOf(classStr(SysUserLogCleanup))]
final class MySysUserLogCleanup_Extension
{
  // static members
  static private SysUserLogCleanup myRunningInstance;

  // Extending class state...
  private boolean myArchive;
  private DialogField myDialogArchive;
  #define.CurrentVersion(1)
  #localmacro.CurrentList
  myArchive
  #endmacro

  public Object dialog()
  {
    Dialog dialog = next dialog();
    myDialogArchive = dialog.addField(extendedtypestr(NoYesId), "Archive");
    myDialogArchive.value(myArchive);
    return dialog;
  }

  public boolean getFromDialog()
  {
    boolean result = next getFromDialog();
    myArchive = myDialogArchive.value();
    return result;
  }

  public void initParmDefault()
```
public void initParmDefault()
{
    next initParmDefault();
    myArchive = true;
}

public void run()
{
    try
    {
        myRunningInstance = this;
        next run();
    }
    finally
    {
        myRunningInstance = null;
    }
}

public container pack()
{
    container packedClass = next pack();
    return SysPackExtensions::appendExtension(packedClass, classStr(MySysUserLogCleanup_Extension),
        this.myPack());
}

private boolean myUnpack(container packedClass)
{
    Integer version = RunBase::getVersion(packedClass);
    switch (version)
    {
        case #CurrentVersion:
            [version, #currentList] = packedClass;
            break;
        default:
            return false;
    }
    return true;
}

private container myPack()
{
    return [#CurrentVersion, #CurrentList];
}

public boolean unpack(container _packedClass)
{
    boolean result = next unpack(_packedClass);
    if (result)
    {
        container myState = SysPackExtensions::findExtension(_packedClass, 
            classStr(MySysUserLogCleanup_Extension));
        //Also unpack the extension
        if (!this.myUnpack(myState))
        {
            result = false;
        }
    }
    return result;
}

private void myArchiveUserLog(SysUserLog _userLog)
{
    if (myArchive)
    {
        //...
    }
// Wire up event handler for deletion of the record
[DataEventHandler(tableStr(SysUserLog), DataEventType::Deleting)]
static public void SysUserLog_onDeleting(Common _sender, DataEventArgs _e)
{
    if (myRunningInstance)
    {
        myRunningInstance.myArchiveUserLog(_sender as SysUserLog);
    }
}
}
In Dynamics AX 2012, there were customization points that allowed you to subscribe to events (Application.Startup delegates) that were raised when the client was initializing. These events were deprecated because there is no concept of a rich client. On the server, only server sessions are considered, however because you can migrate logic from previous releases, new events have been added to the ApplicationStartupEventManager class.

The following sections highlight the new data sources that you can add to existing forms by using extensions.

**static delegate void onSystemStartup()**
- This event occurs when the system starts up.
- It is raised once per AOS upon startup.

**static delegate void onFirstTimeUserInteractiveSessionCreated()**
- This event occurs when the system is creating an interactive session for the first time for a user.
- It is raised once per user per AOS.

**static delegate void onFirstTimeUserNonInteractiveSessionCreated()**
- This event occurs when the system is creating a non-interactive session for the first time for a user.
- It is raised once per user per AOS.

**static delegate void onInteractiveSessionCreated()**
- This event occurs when an interactive session is created and ready for use.
- It is raised once per interactive session creation for any user.

**static delegate void onSessionCreated(boolean _isBatch, boolean _isInteractive)**
- This event occurs when the session is created and ready for use.
- It is raised once per interactive session creation for any user.
- _isBatch specifies whether the system is running a batch job.
- _isInteractive specifies whether the session is interactive.
To modify properties on an existing field in a table, you must first create an extension for the table. You can modify the following properties:

- **Label**
- **Help text**
- **Country Region Codes**
- **Extended Data Type** – You can select only extended data types (EDTs) that are derived from the currently selected EDT. The lookup in the property sheet is filtered so that only those EDTs are shown. For example, to edit the EDT on the **Width** field in the InventTable table, you can create a derived EDT that is based on **BOMMeasureWidth**, and then modify the **Extended Data Type** property on the **Width** field in the InventTable extension. In this way, you can modify the look and feel of the **Width** field in the user interface when the new package is deployed.
To add a new field to an existing table, you must first create a table extension. For example, to add a field that holds the radius of the released product, you must create an extension for the InventTable table in your model, as shown in the following illustration.

You can now add the field to the extension, just as you would add a field to a table in your model. You can use two methods:

- In the designer, right-click the **Fields** node, select **New**, and then select the type of field to add.
- Drag an existing Extended Data Type or Base Enumeration from your project onto the **Fields** node.

When you've finished, you can modify the properties of the new field. In the following illustration, only the **Label** property was modified.

You can now optionally add the new field either to one of the existing field groups or to a new field group that you create. In the following illustration, the **Radius** field was added to the **PhysicalDimensions** field group.
After compilation and synchronization of the database, you can see and edit the new field in the user interface.
Often, you extend tables so that you can store additional data for later but also quickly access the data that is based on the new fields. Therefore, it's often beneficial to have a dedicated index that speeds up the database search. You can add a new index to an existing table through extension. To add an index to an existing table, you extend the selected table and then create an index just as you would create an index on a new table. You can add both new and existing fields so that they are part of the new index.

In the following illustration, an InventTable extension is used to define an index for a new field on the InventTable table.

**WARNING**

You should not use this approach to create unique indexes. This change is an intrusive change that might break the solutions of other independent software vendors (ISVs) if those solutions are deployed in the same environment. This capability will be removed in future platform releases.
To enable rich and secure interactions with data in multiple tables, you must help guarantee referential integrity by defining relations that describe the link between two tables. By defining relations, you enable validation of the data that is entered and lookup capabilities for the related information.

You can add a new relation by extending a table.

In the following example, a new field, `MyInventLocationId`, is added to the InventTable table. This field is a reference to the InventLocation table that contains warehouses.

1. In the new extension model, create an extension of the InventTable table.
2. Create a new relation, just as you would create a relation on a regular table.
3. Specify the Related Table, Relationship Type, and Cardinality properties, and any other properties that apply to the relation.
4. Add the link by specifying the fields from the InventTable table and the InventLocation table that have the same value. In this case, the fields are `MyInventLocationId` in the InventTable table and `InventLocationId` in the InventLocation table.

The following illustration shows the new relation.

![Illustration of the new relation](image)

**Troubleshooting**

**Navigation property methods not working**

**Issue** - Navigation property methods do not work when a foreign key relation is created using a table extension. The compiler will not allow a call to a navigation method on the extended table.

**Solution** - Navigation methods are not supported at this time.
To modify properties on a table, you must create an extension of that table. In Application Explorer, right-click the table, and then select Create extension. A new table extension is created in the selected project, as shown in the following illustration.

You can now modify the following properties through the property sheet:

- Country Region Codes
- Created By
- Created Date Time
- Form Ref
- Modified By
- Modified Date Time
- Preview Part Ref
- Tags
- Title Field1
- Title Field2

By setting the Created By, Created Date Time, Modified By, or Modified Date Time property to Yes, you help guarantee that a corresponding field is added to the table. Corresponding tracking information about the user is then stored in the table when records are created or updated. You can’t set these properties to No if they are set to Yes on the base table.

By adding country or region codes to the list, you help guarantee that the corresponding table is also applicable when the system runs in the context of the specified country or region.
When you extend the business logic that is related to a table, the general coding principles that help keep your
code clean still apply. Therefore, you must eventually encapsulate actions in separate methods on the table. In
Microsoft Dynamics AX 2012, you completed that task by adding the method directly on the table through
overlayering. To complete the same task through extension, you use a different approach. Specifically, you create
an augmentation class.

For example, a new field that is named **MyInventLocationId** was added to the InventTable table through
extension. A data event handler was also created for the **Inserting** event, and you must implement the logic of
filling the new field there. To encapsulate that action, you will create a new method on InventTable and name that
method **myDefaultInventLocationId**.

You first create a new class in the extension model. This class will augment the InventTable table, and enable
access to the table’s fields and methods in a manner that is easy to read and understand. It’s important that you
choose the correct name for your augmentation class. This name must be unique across all types in all models
that are deployed. For more information, see [Naming guidelines for extensions](#).

```csharp
[ExtensionOf(tableStr(InventTable))]
final class InventTableMy_Extension
{
    public void myDefaultInventLocationId()
    {
        // This would have partner specific logic to initialize the new field.
        this.MyInventLocationId = this.inventLocationId();
    }
}
```

You can now add new methods to the augmentation class. These methods will then appear in IntelliSense for
variables of the **InventTable** type, just as if they were defined directly on the table. This behavior applies to both
static methods and instance methods.

There are a few rules for augmentation classes:

- They must be final.
- They must be suffixed by **_Extension**.
- They must be decorated with the `[ExtensionOf()]` attribute.

Now you can use your new method, for example, from an event handler:

```csharp
class InventTableMy_EventHandler
{
    [DataEventHandler(tableStr(InventTable), DataEventType::Inserting)]
    public static void InventTable_onInserting(Common sender, DataEventArgs e)
    {
        InventTable inventTable = sender as InventTable;
        // Call the method as if it was defined directly on InventTable.
        inventTable.myDefaultInventLocationId();
    }
}
```
NOTE

It is common for event handler classes to contain handlers for any number of events. However, it is not good practice to put event handlers in augmentation classes. Doing so makes the event handler methods available as methods on the augmented type. This is incorrect because the event handler is intended to be called through the event, not explicitly as a method on the type.
As part of your business workflows, records are regularly inserted, updated, and deleted. To customize the system behavior, you can hook into some of the record operations that are most often used. For example, you can fill additional fields on the record, perform additional data validation, or insert additional data into related tables. Several events that are available on the table let you achieve those customizations through extensions. Your code can subscribe to these events, and can insert your logic before or after the event is run.

For a list of the events that you can subscribe to, see the “Table extensions” section in Customize through extension and overlerying.
Often, the information that is stored in existing tables doesn't satisfy customer requirements. Therefore, additional tables must be created, and data from those tables must be shown on pages.

You can add new data sources to existing forms through extension. Follow these steps.

1. In the extension model, create a form extension for the selected form.

2. Right-click the form extension, and then select **New Data Source**.

3. Specify the **Table** property and other required properties on the data source. For example, define how the data source should be linked with the other data sources for the form.

4. Drag fields from the new data source into the form design, as shown in the following illustration.

5. In a similar manner, you can add fields from existing data sources. For example, the table behind the form might have been extended with additional fields, as shown in the following illustration.
You might have to right-click the form extension data source and then select **Restore** to make the new fields appear in the list.

6. You can now view and edit the data in these new fields and tables, as shown in the following illustration.
The form caption appears in the page tab next to the web browser’s Address bar and helps the user identify the page that is currently open. In metadata, the form caption is represented by a property on the form design. Therefore, to change the caption, you must modify the **Caption** property on the form design. You can make this change through extension. Create an extension of the selected form in the extension model, and then change the **Caption** property as usual.

The following illustration shows what the form caption looks like in a browser.

![Image of form caption in a browser](https://usnc obe axax1aoc.cloud.on...)

Dynamics 365 | Operations
---|---
Edit | New Delete | PRODUCT | PURCHASE | SELL | MANAGE INVENTORY

Click the edit button to make changes.

**NOTE**

None of the other properties on the form design can be changed.
Often, the way that users interact with the application requires modification. Typically, you hide or disable controls on the page, replace standard labels with labels that are more appropriate, or even add new controls that the user requires. You can also create a form extension.

**TIP**
You can achieve even more flexibility through event subscription on form control events. This approach is discussed in other topics. In this topic, the focus is on metadata changes.

**Example**

The customer requires changes to the Manage inventory FastTab on the Released product details page. You must change the label of the FastTab, disable the field group that shows the catch weight configuration, and add new controls. (For this example, the new controls aren't bound to existing fields in the data source).

Follow these steps to make the required changes.

1. In the extension model, create an extension of the EcoResProductDetailsExtended form.

2. Navigate through the form design tree to the TabPageInventory tab page (`Design > Tab > Details > GroupDetails > TabHeader > TabPageInventory`), select it in the designer, and open the Property sheet.

3. Update the Caption property to the desired value.

4. Right-click the tab page, and then select **New**. Set the required properties on the new control. You can also move the control up and down in the immediate container to position it correctly.

**NOTE**
Alternatively, right-click the subnode that the new control should appear after, select **Insert sibling**, and then select the type of control to add.
Of course, you can just drag bound controls over from the corresponding data source.

5. Select the **PdsCatchWeight** group control, and change the **Enabled** property to **No**.

**NOTE**

If you change metadata properties such as **Enabled** and **Visible**, there is no guarantee that the control will stay in that state at runtime. After a form is loaded, business logic on that form can change the state of controls through code.

When you’ve finished, the page includes additional fields, catch weight information can’t be edited, and the whole FastTab has a different caption.
NOTE

You can't modify the **AutoDeclaration** property on controls. However, you can still access the controls by name from code.
Extend the scope of number sequences

This topic shows you how to extend the number sequence scope.

The scope of a number sequence defines which organization uses the number sequence. The scope can be Shared, Company, Legal entity, or Operating unit. Company and Legal entity scopes can be combined with fiscal calendar periods to create even more specific number sequences. New number sequence scopes can be added through extensions.

To create a new scope and have it show up in the client, complete the following steps:

1. Create an enum extension for NumberSeqParameterType. In the extension, add a new enum value for the new scope type.
2. Create an enum extension for NumberSequenceType. Add a new enum value for the new scope type. The NumberSequenceType enum is used in NumberSequenceTableEntity and NumberSequencesReferenceEntity.
3. Create a table extension for the NumberSequenceScope table. Add a new field for the new scope type.
4. Create an extension class for NumberSeqScope class.
   a. Create a post handler for the NumberSeqScope::getValidScopeTypes method. In the event handler method, add the new scope type to the valid scope types list.
   b. Create an event handler for the onGetFormatSegmentShortName delegate. In the event handler, return the short name for the new scope type.
   c. Create a post handler for the NumberSeqScope::find method and add logic for the new scope type so the corresponding NumberSeqScope instance can be found.
   d. Create a post handler for the NumberSeqScope::getId method and add logic for the new scope type, so the corresponding record can be found (or created if it does not exist) in the NumberSequenceScope table.
5. Create an extension class for the NumberSequenceScopeFactory class. Add a method that initializes the new NumberSeqScope that represents the new scope type.
6. Create a form extension for the NumberSequenceDetails form. Add controls that show the new scope type to the Scope tab.
7. Create an extension class for the NumberSequenceDetails form.
   a. Create a post handler for the updateScopeControlVisibility method to show the new scope type when the new scope type is selected in the Scope box.
   b. Create a post handler for the updateScopeControlValues method to update the values of the controls in the Scope tab.
   c. Create a post handler for the createScope method to initialize a NumberSeqScope instance when the new scope type is selected.
   d. Create an event handler for the getShortNameForParameterType delegate to return the short name for the new scope type.
8. Add an extension class for the NumberSequenceTableEntity and NumberSequencesReferenceEntity data entities. Create post handlers for the GenerateNumberSequenceScopeTypes and GenerateNumberSequenceScopeValues methods to generate the NumberSequenceScope for the new scope type.
This topic provides a high-level overview of how to add new inventory dimensions through extensions. It also includes information about how to access a sample application that contains an actual implementation.

**Solution overview**

The cornerstone in this solution is that multiple roles participate in the life cycle of adding new inventory dimensions through extensions. The following description simplifies and generalizes this solution, however, in real life there is overlap between the roles, and sometimes it might even be the same person filling several roles.

**Microsoft's role**

Microsoft provides a finite set of unused dimension fields.

In addition to the 15 existing dimensions, Microsoft supports 10 generic dimensions:

- 8 string-based
- 1 real-based
- 1 utcdatetime-based

This brings the total number of inventory dimensions in the standard application to 25:

- 5 product dimensions: Color, Size, Style, Config, and Version
- 5 tracking dimensions: Serial, Batch, Owner, Profile (Russia only), and GTD (Russia only)
- 6 storage dimensions: Site, Warehouse, Location, Status, License Plate, and Pallet (for upgrade and migration only)
- 12 unassigned generic dimensions: InventDimension1 to InventDimension12

Microsoft provides the physical schema.

**ISV role**

The ISV adds new inventory dimensions. The ISV solution provides all the specific functionality for the dimension - it must be strong-typed, maintainable, testable, and performant. In addition, the solution must be agnostic to other ISV's solutions. The ISV builds a solution that doesn't reference the physical schema directly, but goes through an indirection, which can be done seamlessly.

The ISV provides the logical implementation.

**VAR role**

The VAR must be able to deliver a fully functional system to a customer. The system can contain solutions from multiple ISV's - each potentially containing new inventory dimensions. In total, up to 10 ISV dimension fields are supported.

The VAR provides the binding between the physical data model and logical implementation.

**Details**

The first half of the solution is straight forward. A new class hierarchy is introduced. Each new dimension must be implemented in a new class deriving from either InventProductDimension or InventTrackingDimension. Currently, there is no support for storage dimensions. With this, ISVs can introduce new dimensions without having to change any of the logic on the InventDim table.
To reference the new dimension in a strongly-typed fashion, the ISV introduces a table extension class to the InventDim table. The extension classes for Style, Color, and Size can be used as templates.

**Example: InventDimStyle_Extension**

```csharp
/// <summary>
/// The &lt;cpp&gt;InventDimStyle_Extension&lt;/cpp&gt; class extends the &lt;cpp&gt;InventDim&lt;/cpp&gt; table with behavior for the style dimension.
/// </summary>
[ExtensionOf(tableStr(InventDim))]
final class InventDimStyle_Extension
{

    public EcoResItemStyleName parmInventStyleId(EcoResItemStyleName _style = this.getValueForDimension(classStr(InventProductDimensionStyle)))
    {
        if (!prmIsDefault(_style))
        {
            this.setValueForDimension(classStr(InventProductDimensionStyle), _style);
        }
        return _style;
    }

    /// <summary>
    /// Returns the field id for the style dimension.
    /// </summary>
    /// <returns>The field id.</returns>
    public static FieldId fieldIdStyle()
    {
        return InventDim::fieldIdForDimension(classStr(InventProductDimensionStyle));
    }
}
```

The dimensions can be referenced like this.
//Setting a value
inventDim.parmISVDim("Some value");

//Select statements
select inventDim
where inventDim.(InventDim::fieldIdISVDim()) == "Some value";

The ISV can now build logic, including the data model and user interface for maintaining the list of dimension values, for the new inventory dimension.

The second half of the solution is the data model. The standard application will contain the following for each new dimension:

- A label file.
- A configuration key.
- Two extended data types (EDTs) (for the field on InventDim and for the flag on InventDimParm).
- One field on InventDim table.
- One field on InventDimParm table.
- One field on InventDimFieldMap map and one field on each of the tables (approximately 30) mapped.

The VAR’s job is to wire the ISV solutions to the available dimension fields on InventDim for a given customer. To minimize this work, it currently includes the following:

- Implement the binding mapping. This is accomplished by extending the method InventDimFieldBinding.className2FieldName().
- Enable the configuration key.
- Extend the EDT to specify the right string size.
- Extend the Label file, such as copy the ISV-provided labels into the correct label file.
- Extend the ProductDimensions or TrackingDimensions field groups on InventDim, and a few other tables, depending on the type of dimension.
- Extend relations and index as needed on InventDim.
There are some technical limitations influencing the design of the solution. The most significant is the SQL statements throughout the application that contain where-clauses on InventDim. Most of these are implemented using macros, which doesn't change the fact that SQL statements are not extensible. Many of the SQL statements could be rewritten to use query objects to make them extensible, however many delete_from and update_recordset would remain. A viable solution cannot require changes to these SQL statements when adding new dimensions.

Another technical limitation is the amount of inventory dimensions that can be supported. Each adds a small overhead, and the InventDimFixed EDT sets an upper limit at 32. This EDT contains a bit mask for each dimension, and because the EDT is an integer, the limit is 32. The provided solution stays within the limit of 32. If required in the future, InventDimFixed could be changed to be an Int64, a container, or it could be removed.

**Sample application**

A sample application called “Product flavor dimension sample app” can be found on [GitHub](https://github.com).

This sample consists of three models:

- ISV production code
- ISV test code
- VAR integration code

Together these models provide a great starting point for implementing new inventory dimensions. The sample application introduces a new product dimension: Flavor.

The application supports many end-to-end business scenarios, for example creating, buying, and selling items with various flavors.

If needed, please log issues directly in GitHub, and feel free to contribute to the sample application to provide additional coverage.
In Microsoft Dynamics 365 for Finance and Operations, Enterprise edition 7.3 and later, the pricing area is extensible. Some common customizations for price and discounts include:

- Adding new price group types and the corresponding price types (enum values for `PriceType` and `PriceGroupType`), in addition to adding search mechanisms for the new price types.
- Modifying the price and discount search, including passing in any additional parameters to the `PriceDisc` class.

**PriceType and PriceGroupType enums**

Typically, adding a new type of price discount search starts with adding a new enum value in the two enums: `PriceType` and `PriceGroupType`. To support extensibility, `PriceType` and `PriceGroupType` enum values are now encapsulated in the class hierarchies `PriceGroupTypeTradeAgreementMapping` and `PriceTypeTradeAgreementMapping`, respectively. These can be extended for any new `PriceType` and `PriceGroupType` extended enum values.

The mapping of fields on the `Customer`, `Vendor`, and `InventTable` tables that correspond to the price types is defined in `PriceTypeTradeAgreementMapping`.

The following diagram highlights the implementation. Note that the methods show only one of the sub-classes. The implementation needs to be on each sub-class.

**PriceDisc class**

The `PriceDisc` class is the search engine for price and discounts. This class uses a `PriceDiscParameters` object as a member for passing in the parameters that are used in the price and discount search. This enables you to pass in the additional search parameters for the specific solutions. Only the parameters for a given `PriceGroupType` search are passed through the corresponding find methods on the `PriceDisc` class.
The ability to wrap and modify the instantiation of the `PriceDiscParameters` class is enabled for all price and discount search calls made throughout AppSuite.

In the following diagram, you can see how the `PriceDisc` class can be extended to modify existing searches or to add new search methods that correspond to the extended `PriceType` enum values.

Add a new price search

In this scenario, you have extended the `PriceGroupType` enum with a new value `PriceGroupTypeISVExtension`, and two corresponding `PriceType` enum values - `ISVPurchPriceType` and `ISVSalesPriceType`.

The following diagram illustrates how a new price search can be added for the `PriceType` and `PriceGroupType` values.
This example shows the following:

- For the newly created `PriceGroupType` value, a `PriceGroupTypeTradeAgreementMappingISVPriceGroupType` class decorated with the attribute `ISVPriceGroupType` defines the behavior of the price group type.
- For the newly created `PriceType` value, the `PriceTypeTradeAgreementMappingISVPurchPriceType` and `PriceTypeTradeAgreementMappingISVSalesPriceType` classes correspond to Purchase and Sales.
- Augmenting the `PriceDiscParameters` class to add any generic parameters for the price discount search.
- Augmenting the `PriceDisc` class to create the new price discount search methods for the new price types.
- The `PriceDiscParameters` is accessible from all classes related to price and discount search and these could be augmented, based on the requirements.
To extend table maps, we have refactored table maps into a model, which allows you to extend a solution with additional fields and methods. This topic discusses why you need a model to extend a table map.

Adding a field to an existing table map through extension can present some challenges. If these issues are not addressed during the implementation, there can be runtime errors. The errors occur because the developer cannot modify all the tables that are involved in implementing the table map. The same is true for adding a method to a table map if the method is called directly as an instance method on the table map. There is no way to enforce how fields on table maps must be mapped to fields on all tables that implement the table map. Similarly, there is no way to enforce how methods on table maps must also be methods on all tables that implement the table map.

The following diagram shows the SalesPurchTable table map, which is implemented by the SalesTable, PurchTable, and SalesBasket tables in the ApplicationSuite model. In addition, the ISV1Header table implements the SalesPurchTable table map, but ISV1Header is part of an ISVModule1 model.

For example, if a new field named AccountingGroupId and a new method named validateAccountingGroup are added to the table map in the ApplicationSuite model, then the tables that you implement the table map can be updated to include the field and method added as well. The ISV1Header table in the ISVModule1 model is, however, outside of the control of the developer making the changes to the ApplicationSuite model.
If you add business logic to the **ApplicationSuite** model, and that logic queries the new **AccountingGroupId** field and the table map record is of type **ISV1Header**, a runtime error occurs.

```java
SalesPurchTable      headerTable;
...                    ...
if (headerTable.AccountingGroupId)
```

Similarly, if you add business logic to the **ApplicationSuite** model, and that logic queries **validateAccountingGroup**, then a runtime error occurs.

```java
SalesPurchTable      headerTable;
...                    ...
if (headerTable.validateAccountingGroup())
```

As a result, the solution is broken, unless you add mapping to the new field and add the new method to the **ISV1Header** table.

The conflict is not resolved if the ability to add fields or methods is added to table maps through extension. This is illustrated in the following diagram, where **ISVModule2** includes extensions of the table map and the implementing tables in the **ApplicationSuite** model. The developer implementing **ISVModule2** has no control over the **ISV1Header** table in the **ISVModule1** model, so the **ISV1Header** table lacks a mapping of the **AccountingGroupId** field and implementation of the **validateAccountingGroup** method.
Even if the compiler enforced that all fields and methods on a table map must be mapped to all tables implementing the table map, the conflict would not be resolved. Instead of receiving runtime errors, adding a field or a method would clear a breaking change, as tables not having a new field mapped or a new method implemented would compile when the model containing the added field/method is applied. To extend table maps, we have refactored table maps into a model, which allows you to extend a solution with additional fields and methods.
The SalesPurchLine and SalesPurchTable table maps expose a set of common fields and methods that are used by a variety of product features. The mapping of fields and the implementation of methods have been refactored into a class hierarchy. Some of these changes include:

- Methods on the table maps have been moved to the class hierarchy.
- Fields are exposed through parm-methods on the class hierarchy.
- The table map still exists and tables still implement the mapping to the table map, but the fields on the table map have been made obsolete, and field mapping has been removed.
- The methods on the table map have been made obsolete.

### SalesPurchTableInterface hierarchy

The new SalesPurchTableInterface class is the abstract base class for the ApplicationSuite functionality that has been introduced by the refactoring of the SalesPurchTable table map. The class contains abstract methods for fields and methods, which must exist for each table implementing the interface. It also contains the default logic in methods, which is common across all implementations. Each table that implements the SalesPurchTable table map must be represented as a derived class in the SalesPurchTableInterface class hierarchy. Each derived class must be decorated with a SalesPurchTableInterfaceFactory attribute class. The attribute is used to associate the derived class with the table, so that it is possible to instantiate a class of the correct type depending on a SalesPurchTable record.
Even though the table map methods have been made obsolete, the corresponding methods still exist on the implementing tables. The logic in these methods have been refactored from delegating calls to the method on the table map, to the corresponding method on the base class of the hierarchy.

**Extension scenario**

In this example, if you want your ISVModule2 model to extend the interface class hierarchy and the tables implementing the `SalesPurchTable` table map, you must perform the following steps:

1. Add the fields and methods through table extension to the necessary tables implementing the `SalesPurchTable` table map.
2. Create a new interface class hierarchy, which exposes the new fields as parm-methods and other additional methods. The base class of the class hierarchy must be concrete and not abstract.
3. Create derived classes for each table that has been extended.
   a. Decorate each derived class with the `SalesPurchTableInterfaceFactory` attribute class to associate the class with the correct table.
   b. Create a static factory method on the base class of the new class hierarchy. The factory method should instantiate the proper derived class that leverages the `SalesPurchTableInterfaceFactory` attribute. If no derived class can be found, then an instance of the base class must be returned.
4. Create an extension class of the `SalesPurchTableInterface` class. The class augments the `SalesPurchTableInterface` class with a method that creates an instance from the new class hierarchy by calling the factory method on the new class hierarchy.

The class and extensions described above are shown in the following diagram.
The diagram contains an ISVModule1 model, which includes the ISV1Header table that implements the SalesPurchTable table map and contains its own SalesPurchTableInterface derived class. The model is independent of the ISVModule2, so when logic in the ISVModule2 creates an instance from the ISV2SalesPurchTableInterface class hierarchy, then an instance of the base class will be returned when the SalesPurchTable record is of type ISV1Header. If the methods on the base class return a reasonable result for unknown tables, then the two ISV models will co-exist within the same installation.
The following code example demonstrates a way to extend table maps.

```java
[ExtensionOf(classStr(SalesPurchTableInterface))]
final public class ISV2SalesPurchTableInterface_Extension
{
    private ISV2SalesPurchTableInterface ISV2SalesPurchTableInterface;

    public ISV2SalesPurchTableInterface ISV2SalesPurchTableInterface()
    {
        if (!ISV2SalesPurchTableInterface)
        {
            ISV2SalesPurchTableInterface = ISV2SalesPurchTableInterface::createInstance(this);
        }
        return ISV2SalesPurchTableInterface;
    }

    public ISV2SalesPurchTableInterface ISV2SalesPurchTableInterface
    {
        SalesPurchTableInterface salesPurchTableInterface;

        private void initializeSalesPurchTableInterface(SalesPurchTableInterface _salesPurchTableInterface)
        {
            salesPurchTableInterface = _salesPurchTableInterface;
        }

        public SalesPurchTable parmSalesPurchTable()
        {
            return salesPurchTableInterface.parmSalesPurchTable();
        }

        protected void new()
        {
        }

        public static ISV2SalesPurchTableInterface createInstance(SalesPurchTableInterface _salesPurchTableInterface)
        {
            SalesPurchTableInterfaceFactoryAttribute attr =
                new SalesPurchTableInterfaceFactoryAttribute(
                    tableId2Name(_salesPurchTableInterface.parmSalesPurchTable().tableId));

            ISV2SalesPurchTableInterface instance =
                SysExtensionAppClassFactory::getClassFromSysAttribute(classStr(ISV2SalesPurchTableInterface), attr)
                as ISV2SalesPurchTableInterface;

            instance.initializeSalesPurchTableInterface(_salesPurchTableInterface);

            return instance;
        }

        public AccountingGroupId parmAccountingGroupId()
        {
            return ''; 
        }
    }
}

[SalesPurchTableInterfaceFactoryAttribute(tableStr(SalesTable))]
public class ISV2SalesTableSalesPurchTable extends ISV2SalesPurchTableInterface
{
    private SalesTable parmSalesTable()
    {
    }
```
return this.parmSalesPurchTable();
}

public AccountingGroupId parmAccountingGroupId()
{
    return this.parmSalesTable().AccountingGroupId;
}

}

[SalesPurchTableInterfaceFactoryAttribute(tableStr(PurchTable))]
public class ISV2PurchTableSalesPurchTable extends ISV2SalesPurchTableInterface
{

    private PurchTable parmPurchTable()
    {
        return this.parmSalesPurchTable();
    }

    public AccountingGroupId parmAccountingGroupId()
    {
        return this.parmPurchTable().AccountingGroupId;
    }

}
PurchLineMap table map logic

When new fields are added to the PurchLine and PurchLineHistory tables using table extensions, the new fields must be copied between the tables when a purchase order is versioned. The PurchLineMap table map specifies the fields that must be copied between the PurchLine table and the PurchLineHistory table when a new purchase order version is created or edited. To accomplish this, extend the PurchLineMap map table to include the additional fields. Additionally, the PurchLineMap is used by the VersioningPurchaseOrder class when archiving purchase order lines. The model is shown in the following diagram.

To be able to specify new fields to be copied, the PurchLineMap table map logic and its usage have been refactored. The copy logic has been moved to the PurchLineVersioning class, so the VersioningPurchaseOrder class references the PurchLineVersioning class instead of the PurchLineMap table map. The PurchLineVersioning class delegates the logic to copy the fields and the logic to determine whether a confirmation is required from the classes that implement the PurchLineIVersioningFieldSet interface. Each class that implements the interface is associated with a table map that specifies the fields to copy.

The PurchLineDictVersioning class instantiates the PurchLineIVersioningFieldSet object using reflection. The PurchLineDictVersioning class collects the entire set of fields which need to be copied. The field data is collected based on all the table maps associated with a class that implements PurchLineIVersioningFieldSet. The following diagram displays the new classes and their dependencies.
How to extend PurchLine and PurchLineHistory tables with new fields

Suppose that you want the ISVModule2 model to extend the PurchLine and PurchLineHistory tables with new fields that must be copied when creating a new version of a purchase order.

**NOTE**

You must have developer access to the ISVModule2 model.

To complete this task, you must perform the following steps:

1. Add fields by using table extensions to the PurchLine and PurchLineHistory tables.
2. Create a new table map containing the fields that must be copied, and implement the new table map on the two new table extensions.
3. Create a new class to implement the PurchLineIVersioningFieldSet interface and implement the following required methods.
   - `copyVersion` method - Copies data between two records of the new table map type.
   - `fieldSetTableMapId` method - Returns the ID of the new table map.
   - `isChangeConfirmationRequired` method - Returns true or false based on whether the change to the newly added field values requires a confirmation to be created.

The classes, interfaces, and extensions described in these steps are shown in the following diagram.
This topic describes how to extend the number of decimals for selected data types. You can create extensions of specific extended data types of the type Real, to change the number of decimals for certain scenarios. To change the number of decimals, change the `NoOfDecimals` property as needed.

Extended data types are hierarchical and inherit behavior from the data type they extend. When changing the number of decimals for one extended data type, the number of decimals on all derived extended data types will follow. In other words, if you find an extended data type where `NoOfDecimalsIsExtensible` is false, then check the parent extended data type, as the number of decimals might be extensible in this wider scope.

Due to database constraints, each of the data types described in this topic can have a maximum precision of six decimals.

### Weight

Weight data can be maintained with a maximum of two decimals by default.

If you require the ability to enter, maintain, and view weight data with a maximum of six decimals, you must extend the number of decimals for the `WeightBase` extended data type.

### Product width, height and depth

These physical dimensions can be maintained with a maximum of two decimals by default.

If you require the ability to enter, maintain, and view this data with a maximum of six decimals, you must extend the number of decimals for the `InventWidth`, `InventHeight`, and `InventDepth` extended data types respectively.

### Product quantity

Quantity data that is related to the procuring, consuming, producing, storing, and selling of products can be maintained with a maximum of two decimals by default.

If you require the ability to enter, maintain, and view product quantities with a maximum of six decimals, you must extend the number of decimals of the `ProductQuantity`, `CostQuantity`, and `CAMMagnitude` extended data types.

Bill of materials, formulas, and production orders allow maintaining quantities with four decimals by default.

If you require more than four decimals, extend the number of decimals for the `BOMProductQuantity` extended data type.

### Related data types

Price unit, Price quantity, and Charge quantity data can be extended independently from product quantities.

You can extend the `PriceUnit` extended data type to change the number of decimals to a value other than the
You can extend the `PriceQty` extended data type to change the number of decimals to a value other than the default two for price and charge quantities.

**Overloaded data types**

There are two extended data types that are used for storing both quantity data and other types of data. These data types must be extended separately.

The `AmountQty` extended data type is used for storing and presenting both amounts and quantities. The `AmountQty` extended data type should be extended to the maximum required number of decimals for both amounts and quantities.

For example, if amounts need to be maintained with three decimals, but quantities still need to be maintained with two decimals, then the data type should be extended to three decimals.

The `ProductQuantityHourValue` extended data type is used for storing and presenting both hours and quantities. The `ProductQuantityHourValue` extended data type should be extended to the maximum required number of decimals for both hours and quantities.

For example, if quantities need to be maintained with four decimals, but hours still need to be maintained with two decimals, then the data type should be extended to four decimals.

**Unit amounts**

By default, unit amounts including prices, line discount amounts, and line charge amounts can be maintained with a maximum of two decimals.

If you require the ability to enter, maintain, and view unit amounts with a maximum of six decimals, you must extend the number of decimals of the `UnitAmountCur`, `UnitAmountMST`, and `CostPriceNonMonetary` extended data types.

If you require a more than four number of decimals, you should also extend the `PriceRoundOff` extended data type.

**Overloaded data types**

There are five extended data types that are used for storing both unit amount data and other types of data.

The `PriceDiscAmount` extended data type is used for storing and presenting amounts and unit amounts. The `PriceDiscAmount` extended data type should be extended to the maximum required number of decimals for both amounts and unit amounts.

For example, if amounts need to be maintained with three decimals, but unit amounts need to be maintained with four decimals, the data type should be extended to four decimals.

The `MCRRoyaltyValue`, `PdsRebateValue`, `TAMRebateValue`, and `MarkupValue` extended data types are used for storing and presenting amounts, unit amounts, and percentages.

The extended data types should be extended to the maximum required number of decimals for amounts, unit amounts, and percentages. For example, if amounts need to be maintained with three decimals, but unit amounts need to be maintained with four decimals and percentages should remain maintained with two decimals, then the data type should be extended to four number of decimals.

**Amounts**

Amounts, including unit amounts, can be maintained with a maximum of two decimals by default.

If you require the ability to enter, maintain, and view amounts including unit amounts with a of maximum six
decimals, you must extend the number decimals of the Amount, AmountMST, and CostAmountNonMonetary extended data types.

If you require a different number of decimals for unit amounts other than for amount, follow the description for how to extend the number of decimals for unit amounts.

**Overloaded data types**

There are three extended data types that are used for storing amount data and other types of data. This means that they must be extended separately.

The AmountQty extended data type is used for storing and presenting amounts and quantities. The AmountQty extended data type should be extended to the maximum required number of decimals for both amounts and quantities.

For example, if amounts need to be maintained with three decimals, but quantities still need to be maintained with two, then the data type should be extended to three decimals.

The PriceDiscAmount extended data type is used for storing and presenting amounts and unit amounts. The PriceDiscAmount extended data type should be extended to the maximum required number of decimals for amounts and unit amounts.

For example, if amounts need to be maintained with three decimals, but unit amounts need to be maintained with four decimals, then the data type should be extended to four decimals.

The MarkupValue extended data type is used for storing and presenting amounts, unit amounts, and percentages.

The extended data types should be extended to the maximum required number of decimals for amounts, unit amounts, and percentages.

For example, if amounts need to be maintained with three decimals, unit amounts need to be maintained with four decimals, and percentages should remain with two decimals, then the data type should be extended to four decimals.
X++ and the metadata model provide a powerful foundation for building business solutions. One of the design pillars is to automate as many technical concerns as possible, so that the engineer can focus on the business domain. For example, if you put all the text resources in label files, one technical concern that you don’t have to worry about is the localization of text resources.

Extensibility is another technical concern. You want other people to be able to extend your solution in a safe, robust, and maintainable way. By default, your solution is highly extensible. However, there are a few guidelines that you should follow to help guarantee completeness.

Responsibilities

Any Finance and Operations environment runs a business solution that includes components from many sources. Typically, each solution has code from Microsoft, independent software vendors (ISVs) and partners, and also internally developed code. Each contributor is responsible for its own contribution to the solution and for the way that its contribution interacts with other contributions.

When you write extensible code, you invite other people to interact with your solution. Your responsibility is to enable other people to be good guests. Here is how you meet this responsibility:

- Make robust extension points – Extension points are the foundation of extenders’ solutions. They must be well-defined and robust from release to release.
- Invite side-by-side customizations – Recognize that multiple extenders might use the same extension point. Enable 1:n (one-to-many) interactions instead of 1:1 (one-to-one) interactions.
- Trust that extenders will be well-behaved – All responsible parties share the same goal: to create great, lasting solutions for the customer. When you create extension points, you give up control and share the responsibility with other people. Assume that extenders will be cautious and use your extension points as they were intended.

Proven principles

All the good engineering practices that you’re already using still apply. Everything that you’ve learned still applies. You don’t have to learn new principles or unlearn old practices. This topic is just highlighting three principles of software craftsmanship that have been sought and taught for decades. These principles not only make your code easier to read, maintain, test, review, and refactor, but also make your code easier to extend. Apply and advocate these principles.

Extend code by using the SOLID principles

SOLID is an acronym for five principles that you can use to make your code easier to extend:

- Single responsibility – Classes and method should have a single responsibility and should not have side-effects. By following this principle, you help guarantee that extension points that are automatically created on public and protected methods will be great extension points.
- Open/closed

  - Open for extension – Open your solution for extensions by designing and considering the extension surface. After an extension point is made available, you’re responsible for maintaining it. This responsibility adds significant restrictions to future development. It’s often preferable to open a
solution up for extension by demand. For example, use internal methods over public methods or private methods over protected methods. Limit your extension surface by making your properties private, and your methods either private or final-protected. In this way, no one can take advantage of a dependency on your logic, either through inheritance or extension.

- **Closed for modification** – Make your logic support extensions without requiring further modifications.

- **Liskov substitution** – Derived classes must be able to be substituted for their base classes. For example, this substitution can be done by providing factories, by using SysExtension, and by using simple construct methods.

- **Interface segregation** – Create concise interfaces. This principle lets extenders provide replacement implementations and is particularly valuable when it’s used together with the next SOLID principle, dependency inversion.

- **Dependency inversion** – Depend on abstractions, not concretions. This principle enables decoupling and lets extenders provide concrete instances that conform to the abstraction that your logic depends on.

**Write clean code**

Clean code can be read like an article. The name of a method provides the heading of the article. The body of the method comes next and really consists of just a few lines of summary. This summary calls a few other methods that have good descriptive names. In this way, the reader can keep exploring details and can also stop at any time without missing any conceptual information.

When code is written like in this manner, methods are short, often less than 5 to 10 code lines. Additionally, the number of parameters is low, often less than two, and the conditions and blocks of code are always a single code line.

Here is an example.

```csharp
public void processOrder(SalesOrder _salesOrder)
{
    if (this.approveOrder(_salesOrder))
    {
        this.confirmOrder(_salesOrder);
    }
    else
    {
        this.rejectOrder(_salesOrder);
    }
}
```

In X++, every protected and public method is an extension point. By writing clean code, you automatically produce extensible code. In the previous example, an extender can change how approval, confirmation, and rejection are implemented. If the implementations had been inline, the code would not be extensible.

**Don't repeat yourself (DRY)**

To help prevent misalignment of implementations, avoid redundancy in your logic. This principle is especially important in the case of extensible code, because the extender might not extend all required pieces and might therefore unintentionally leave the solution broken.

**Best practices to create an extensible solution in X++**

The following best practices can help you create extensible solutions in X++, so that consumers of your code can extend your solution:

- **Classes**
Breaking changes

When you make your solution extensible, you also help guarantee that you won’t break extension points later. For more information, see Breaking changes.

External resources

The following external resources can help you make sure that you’re writing clean code:

- SOLID Principles
- Pluralsight - Clean Code: Writing Code for Humans
- Clean Code: A Handbook of Agile Software Craftsmanship
- Clean Coders
- Don't repeat yourself
A class and its methods should have a single responsibility. Keep the following in mind in order to design classes that are resilient to changes in the long run. Class should have:

- A clear purpose
- Good names (class name and method names)
- Only methods that should be extended are exposed for extensibility, i.e. the key rule is allow extending as little as necessary.
  - Every public and protected method is an extensibility point in X++. Every time that a new method is introduced, downstream consumers get a new way to inject additional logic into the method.

**Example**

**Non-extensible code**

```java
void calculatePrice(SalesLine _saleLine, AmountMST _amount)
{
    // cannot add extra condition if needed
    if(_saleLine.QtyOrdered > 0 && _saleLine.SalesType == SalesType::Sales)
    {
        ttsbegin;
        // calculation of SalesPrice is locked and cannot be extended
        _saleLine.SalesPrice = _saleLine.QtyOrdered * _amount;
        _saleLine.update();
        ttscommit;
    }
}
```

**Extensible code**

```java
protected boolean canUpdateSalesPrice(SalesLine _saleLine)
{
    return (_saleLine.QtyOrdered > 0 && _saleLine.SalesType == SalesType::Sales);
}

protected SalesPrice calculateSalesPrice(SalesLine _saleLine, AmountMST _amount)
{
    return _saleLine.QtyOrdered * _amount;
}

public void updateSalesPrice(SalesLine _saleLine, AmountMST _amount)
{
    // extra condition can be added in CoC on the method
    if(this.canUpdateSalesPrice(_saleLine))
    {
        ttsbegin;
        // extra calculation/value can be added in CoC on the method
        _saleLine.SalesPrice = this.calculateSalesPrice(_saleLine, _amount);
        _saleLine.update();
        ttscommit;
    }
}
```
Class hierarchies

For new or existing class hierarchies where a factory pattern can be used, the SysExtension framework enables easy extensions. Because these extensions are truly decoupled, new subclasses can be added without requiring any changes to the base class. Therefore, less code is required. Additionally, because the contract of the construct remains the same, there is no change to the public application programming interface (API). Therefore, refactoring involves low risk.

Some existing factory methods might not instantiate subclasses by using the instance constructor. Instead, they might call a static constructor, such as `construct`. If the SysExtension framework is used for these factory methods, a breaking change occurs, because the static constructors on the subclasses are no longer invoked. In these situations, use the SysExtension framework only for the default case.

For more information about class hierarchies, see the following blog posts:

- SysExtension Framework – to the rescue
- Embrace the extensions mindset with Dynamics 365 for Finance and Operations #2 – SysExtension framework

Deprecation

If a class or a public or protected method is no longer required, always use a warning first to notify consumers that the method is obsolete. Then, when all consumers have had the chance to uptake the changes or the new API, the method can be deprecated. Deprecation of classes and methods (or removal of class members in other cases) is a breaking change. For more information, see Breaking changes.
Before you make a method extensible, you should assess the exposed functionality of the method and the impact that the extensions might have on the scenario where the method is used. For example, depending on the business scenario, there is low risk if you enable extensions to initialize a table record but high risk if you enable extensions to skip a specific validation. You might also want to consider the impact if the method is extended in parallel with other extensions.

After you’ve made a method extensible, future modifications to the method are restricted because of the potential user impact if the method signature or logic is changed.

Here are some guidelines to follow when you write extensible code:

- **Write short and concise methods** – A method should have only one responsibility. This approach enables easy extensions of the method, where the extensions can act only on the specific responsibility of the method. As a simple example, keep the construction and initialization of a class object in two separate methods.

- **Expose only what is necessary** – For any new class members or methods that are added, ‘Keep any new class members or methods that you add private, to allow minimal access to them.

- **Use private, protected, public, and final explicitly** – For methods and class fields, this approach will guide any extenders of your code to your extension points but still let you keep full control of the parts that the extenders should not care about or depend on.

**Method parameters**

- The method is most likely long and should be refactored. Consider whether you should refactor the whole method into a class or split the method into smaller methods that require fewer parameters.
- In other cases, when several parameters are required, the parameters often have a coherence that can be expressed by a class. By encapsulating these parameters in a class, you make it easy for extenders to add additional parameters to the base method, without breaking application programming interfaces (APIs) later.

- **Switch blocks**

  - Avoid switch blocks in the middle of methods. A switch block should be in its own method to enable it to be extended.
  - **Long case blocks** are good candidates for being refactored into a class/class hierarchy that has a subclass for each case block. For an example, see the SalesLineCopyFromSource class hierarchy.
  - Avoid **default blocks** in switch statements, because they make the method that has the switch block non-extensible.
  - Avoid **throw statements in the default block** of a switch statement, because they make the switch statement non-extensible. One way to handle the throw in the default case is to refactor the switch block to a separate method that is extensible. Alternatively, you can make the whole method replaceable.

In the following example, `findOrderHeader` is replaceable.
private Common findOrderHeader(boolean _forUpdate) {
  switch (this.InventTransType) {
    case InventTransType::Sales:
      return this.salesTable(_forUpdate);
    default:
      return this.findOrderHeaderDefault(_forUpdate);
  }
}

protected Common findOrderHeaderDefault(boolean _forUpdate) {
  throw Error::wrongUseOfFunction(funcName());
}

**While** – Avoid `while` blocks in the middle of methods, because it becomes more difficult to extend the `while` blocks. Ideally, logic in a `while` block should be in a separate method that enables extensions.

**Refactoring logic within a while loop**

```java
while select forupdate markupTransOrig
  where markUpTransOrig.TransId == origLineUpdate.TableId
    && markUpTransOrig.TransRecId == origLineUpdate.RecId

  markUpTransOrig.clear();
  markUpTransOrig.initFromSalesLine(_newLine);
  markUpTransOrig.initFromMarkupTrans(markUpTransOrig);

  markUpTransOrig.LineNum = markUpTransOrig.LastLineNum(_newLine.TableId, _newLine.RecId) + 1;
  markUpTransOrig.OrigRecId = origLineUpdate.RecId;
  markUpTransOrig.OrigTableId = origLineUpdate.TableId;
  markUpTransOrig = this.initializeNewMarkupTrans(_origLine, _newLine, markUpTransOrig);

  if (markUpTransOrig.MarkupCategory == MarkupCategory::Fixed)
    markUpTransOrig.Value = CurrencyExchangeHelper::price(markUpTransOrig.Value, markUpTransOrig.CurrencyCode);
    markUpTransOrig.Value = markUpTransOrig.Value;
    markUpTransOrig.update();
  else
    markUpTransOrig.Value = markUpTransOrig.Value;
```

**Extensible method after refactoring**

```java
protected MarkupTrans initializeNewMarkupTrans(SalesLine _origLine, SalesLine _newLine, MarkupTrans _markupTransOrig)
{
  MarkupTrans markupTransNew;
  markupTransNew.clear();
  markupTransNew.initFromSalesLine(_newLine);
  markupTransNew.initFromMarkupTrans(_markupTransOrig);

  markupTransNew.LineNum = markupTransOrig.LastLineNum(_newLine.TableId, _newLine.RecId) + 1;
  markupTransNew.OrigRecId = _markupTransOrig.TransRecId;
  markupTransNew.OrigTableId = _markupTransOrig.TransTableId;

  if (_markupTransOrig.MarkupCategory == MarkupCategory::Fixed)
    _markupTransOrig.Value = markUpTransNew.Value;
  else
    markupTransNew.Value = _markupTransOrig.Value;

  return markupTransNew;
}
```

**If..else statements**
- To enable extension of the conditions in an if statement, extract the logic in the if condition into a separate method.
- Avoid nested if..else blocks, because they make it difficult to change the logic in one of the blocks. One way to resolve this issue is to refactor each condition and the logic in each block into a separate method. In this way, you can extend the conditions or the logic in each block.
- When the if..else blocks handle specialization, consider moving the logic into a class hierarchy. For an example, see `SalesLineCopyFromSource`.
- In some scenarios, a throw in an 'else' block of a method (when the method only has an if..else) makes the method non-extensible. One way to handle the throw in the else is to refactor the conditions for the throw into a separate method.

- **Avoid using PrmIsDefault** – When the method is overridden or wrappable, the caller of `super()` or `next()` provides all parameters. Therefore, `prmIsDefault()` always returns false.

- **Avoid using enumCnt** – At compile time, this method uses a numeric literal of the number of values that an enum has. If the enum is extended or made extensible later, your code will have to be recompiled. Use `DictEnum.values()` instead.

- **Construct methods**
  - Use the `SysExtension` framework to enable easy extensions.
  - Avoid a throw in factory methods. One way to resolve this issue is to extract the conditions for the throw into a separate method that is extensible. For more details, see the guidelines for throw statements later in this list.

- **Static methods** – Static methods can't be extended with extra state. For example, a method extender can introduce properties that can be set by using parameter methods. Use instance methods instead, whenever this approach is possible.

- **Ability to extend part of the logic in a long method** – If it isn't possible to refactor a whole method, but the goal is to make part of the method extensible, apply the extract method refactoring. The new protected method must have a single responsibility, and it must also have a name that conceptually and precisely describes that responsibility. In this way, owners and all extenders can use the method without breaking each other. For example, initialization, insertion, updates to a table record, or instantiation and initialization of a class can be extracted into smaller methods, and each of these smaller methods can be enabled for extensions. The original method then calls these individual methods. Therefore, the callers to this method aren't broken.

- **Throw statements** – A throw that is added to an existing method that is extensible could break extenders. Consider adding the conditions for the throw in an extensible method. In this way, extenders can take advantage of the method, and you can get rid of the throw.

### If condition refactored out to a protected method

```java
if (this.lineDiscountAmount() && accountDisc)
```

Extensible method after refactoring

```java
protected boolean mustPostLineAccountDiscount(SalesPurchLine _salesPurchLine, LedgerDimensionDefaultAccount _accountDisc)
{
    return (this.lineDiscountAmount() && _accountDisc);
}
```

- **Create, read, update and delete (CRUD) statements**
- Use Query objects in scenarios where the queries should be extensible. Implement a protected method that builds the query. In addition, you might want to build several separate methods to add joined data sources, ranges, and selection fields. In this way, different parts of the query can be extended individually.
- Use `SysQueryInsertRecordSet` to convert insert_recordset to a query.
- Avoid field lists in select statements. In this way, you enable extenders to retrieve their additional fields without having to extend.
- Use the `in` keyword in query ranges to enable extenders to add more values to the query range. We recommend this approach especially for query ranges that have enum values.
Methods on forms

- In general, the guidelines for writing extensible methods also apply to form methods.
- Chain of Command (CoC) gives access to the form’s non-private members, which are the same as the non-private members for classes.
- CoC is enabled for nested classes. Therefore, methods that are defined in various levels on the form are extensible.

**NOTE**

One limitation of methods on forms, that for form data source methods only methods that are defined in the kernel are enabled for extensions, i.e. methods defined on the form data source are not extensible. This will be available in an upcoming Platform Update.

Field groups

Consider using field groups whenever possible. In this way, independent software vendors (ISVs) can add their fields for free when they extend a field group.

Form controls

Moving around form controls could potentially cause a break if the controls are made non-extensible by moving.
Extended data types (EDTs) have a rich extension model that lets extenders change specific behaviors.

To provide an extensible solution, keep the following guidelines in mind when you work with EDTs.

**Label/Help text**

Labels and Help text properties can be changed by an extension, but only one value can remain. If multiple solutions change the label of the same EDT, the various labels are, in functional terms, mutually exclusive. Therefore, those labels can't all be installed on the same system.

**String size**

String size can be defined only on root EDTs. The system will use the largest value that is defined across the EDT and its extensions.

For derived EDTs, string size can't be changed by an extension, because the IS-A relationship between the EDTs will be broken.

Assignments to string EDTs will truncate the string to match the defined string size.

**Extends**

The `extends` property can't be changed by an extension. Any change that is made to this property after release will cause a breaking change. Therefore, you must make sure that the property is set correctly before release.

If you set this property, neither you nor extenders will be able to make changes to the string size later.

Avoid unnecessary dependencies. For example, don't extend generic EDTs such as Name and Description.

**Number of decimals**

The `Number of decimals` property can't be changed by an extension.

If you set this property to `True`, extenders can change the number of decimal places.

If you set this property to `True`, make sure that the following conditions are met:

- All truncation logic honors the number of decimal places that is specified on the EDT, so that no implicit or hardcoded rounding will occur.
- The value isn't assigned to other incompatible EDTs that don't correctly handle rounding.
An enumeration (enum) is made extensible by setting the following enum properties:

- Is Extensible = True
- UseEnumValue = No

If you set these properties, downstream implementors can extend the enum with more elements. The values of the elements are determined at deployment time and won’t be identical across systems. However, the following behavior is ensured:

- Data upgrade scripts aren’t required. Enum values are persisted in the database, regardless of the enum that is extended. Therefore, when an enum is made extensible, the enum values that are used on any system will prevail.
- The first element in the enum gets a value of 0 (zero). Therefore, an extensible enum can still be used with the `not` operator. The only exception is when the first element of the enum had a non-zero value before the enum was made extensible.

### Using extensible enums in code

Because enum values are no longer controlled by the developer, there is no certainty about the enum values. When you use extensible enums in code, remember that extensible enums can’t be used in comparisons. For example, `MyEnum::Value1 > MyEnum::Value2`.

Also, look for any conversions between integers and enums. For example, modeled ranges in views and queries, and queries that are created from code by using comparisons, such as `<` and `>` or by using hardcoded integer values in comparisons.

When the model and all dependent models are compiled, the comparisons and conversions to integers will be detected by the compiler as errors.

Make sure that logic where the enum values are used is extracted in smaller methods. In that way, an extension that uses Chain of Command (CoC) can handle the enum values that are added.

For **construct** methods where the instantiation is based on enum values, replace switch blocks with **SysExtension** wherever such a replacement is possible. In other cases, make sure that the default block is extensible. For an example, see the **PurchRFQCaseCopying** class.

If the enum is used in **switch blocks**, avoid having default blocks that either have or don’t have throws that aren’t extensible. When there are **long switch case blocks** or **if...else blocks** for the enum values, consider creating a class hierarchy to handle specific logic that is related to the enum. For an example, see the **PriceGroupTypeTradeAgreementMapping** class hierarchy.

Use the `in` keyword for query ranges that use the enum values, and make the container that the `in` keyword uses extensible.

### Potential issues

Some enums require the elements to have a certain order or value, and cannot be made extensible. This could be status enums, where the values represent a logical progressive sequence, like: Draft, Approved, Completed, or Archived. It could also be enums where the values must have a fixed integral value to match another artifact, like
another enum or a tabpage control's number.

Some enums have many elements. Enums support up to 250 elements. If your enum has many elements, such as more than 100, consider redesigning the solution instead of making the enum extensible. If the enum is extensible, then adding more elements in the future might break customers’s combined solution as the addition might exceed the limit.
Although you can subscribe to existing delegates, don't create new delegates. The Chain of Command (CoC) provides a richer, more robust, and more concise extension mechanism that supersedes delegates.

Instead of creating new delegates, structure your code in small methods that have good names, as described in the guidelines for writing extensible methods.

If you decide to use delegates, consider ensuring no more than one response where applicable. For more information, see EventHandlerResult classes in request or response scenarios.
Tables have a rich extension model that lets extenders add fields, field groups, indexes, relations, methods, and more.

**Unique indexes**

Unique indexes can't be changed by an extension. Unique indexes define a table constraint, and they often also define the key of the rows in the tables. You aren't allowed to change unique indexes, because such changes change the nature of the table. Therefore, there is a high risk that the changes will cause logical conflicts with future versions of the solution that defines the table, or with other solutions that consume the table.

Avoid unique indexes that are likely to be changed either now or in the future. For example, don't create a unique index on product dimensions such as color, size, style, and configuration. Instead, create a unique index on a distinct product variant, so that the index doesn't have to be changed if new product dimensions are added.

If you require an extensible uniqueness constraint on multiple columns, consider creating a hash of the column's values. For an example, see the NumberSequenceScope table.

**Data events**

Tables have many predefined data events that are automatically raised.

Avoid calling `doInsert()`, `doUpdate()`, and `doDelete()`. These methods prevent the data events from being raised and make your table harder to extend. Instead, call `insert()`, `update()`, and `delete()`.

**Field groups**

Always use field groups to group related fields, and to build forms and reports. By consistently using this approach, you enable the extension to surface additional fields in forms and on reports by extending the field group.
This topic describes the various attributes that can be used to control extensibility capabilities for methods.

The following table provides an overview of the default support for extensibility and accessibility on methods. The table also provides guidance on the method signature changes.

<table>
<thead>
<tr>
<th>ATTRIBUTE</th>
<th>HOOKABLE</th>
<th>WRAPPABLE</th>
<th>REPLACEABLE</th>
<th>ACCESSIBILITY</th>
<th>SIGNATURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>private</td>
<td>No</td>
<td>N/A</td>
<td>N/A</td>
<td>Accessible from within class it is defined in.</td>
<td>Signature can be changed</td>
</tr>
<tr>
<td>protected</td>
<td>No</td>
<td>Yes, from Platform update 25 onward</td>
<td>No</td>
<td>Accessible from with the class it is defined, from derived classes, and from classes in the same model.</td>
<td>Signature must remain compatible</td>
</tr>
<tr>
<td>internal</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Accessible in the same model.</td>
<td>Signature can be changed</td>
</tr>
<tr>
<td>protected</td>
<td>No</td>
<td>Yes, unless marked final</td>
<td>No</td>
<td>Accessible from with the class it is defined and from derived classes.</td>
<td>Signature must remain compatible</td>
</tr>
<tr>
<td>public</td>
<td>Yes</td>
<td>Yes, unless marked final</td>
<td>No</td>
<td>Accessible from within the class it is defined, derived classes, and other classes that have access to the defining class.</td>
<td>Signature must remain compatible</td>
</tr>
</tbody>
</table>

**Hookable**

If a method is hookable, extenders can subscribe to pre-events and post-events.

For public methods, you can opt out by adding `[Hookable(false)]` to the method.

You can opt in for private and protected methods by adding `[Hookable(true)]` to the method.

If a method is explicitly marked as `[Hookable(false)]`, then it is not wrappable.

**Best practices when you write code**

When a method is hookable, the compiler generates extra intermediate language (IL) code to enable the method as an extension point. Although the extra code has performance overhead, this overhead is negligible in most cases. However, for performance-critical methods, consider marking the method as non-hookable.
Wrappable

If a method is wrappable, extenders can wrap it by using Chain of Command (CoC). Extenders must call next, because they aren't allowed to break the CoC.

For protected and public methods, you can opt out by adding `[Wrappable(false)]` to the method.

You can't opt in for private methods.

Best practices when you write code

CoC resembles inheritance in many ways. Typically, if you want other people to be able to call your method but not change it, you mark the method as final. Consider marking these methods as non-wrappable or non-hookable.

Replaceable

If a method is replaceable, extenders can wrap it by using CoC, but they don't have to unconditionally call next. Although extenders can break the CoC, the expectation is that they will only conditionally break it. The compiler doesn't enforce calls to next.

To be replaceable, a method must also be wrappable.

For wrappable methods, you can opt in by adding `[Replaceable]` to the method.

Best practices when you write code

When a method is replaceable, it can be extended by using CoC, and the execution of next can be skipped. Before you enable a method to be replaceable, you should thoroughly assess the functional impact if an extender skips the execution of the method.

- **Do** make sure that methods that have `[Replaceable]` have XML documentation that describes the responsibility of the method.

- **Don't** use `[Replaceable]` to let consumers skip the replaced logic and do nothing.

- **Don't** use `[Replaceable]` for factory methods when `SysExtension` can be used instead.

- **Avoid** using `[Replaceable]` when the method changes databases or class state.

- **Avoid** using `[Replaceable]` if the method performs multiple operations and has multiple responsibilities. Instead, refactor the method into separate methods, each of which has a single responsibility, and consider which methods should actually be replaceable.

- **Consider** using `[Replaceable]` to solve transformations.

  Example: Enum conversion that uses a switch statement over enum values, where the default block has a throw.

- **Consider** using `[Replaceable]` to override lookups and jumprefs.

Best practices for extenders

- **Don’t** write logic that has a different responsibility than the logic that is being replaced.

- **Do** call the base functionality (call next) when the replacement logic doesn’t apply.

- **Avoid** replacing logic completely by not calling the base functionality (call next).
When you make your solution extensible, you also help guarantee that you won’t break the extension points later. A breaking change is any change that can break a consumer of your code.

This topic lists some of the types of changes that can break your code.

**IMPORTANT**

This list isn’t exhaustive. Other types of changes that aren’t listed here could also be breaking changes.

## Data model changes

### General

If any data model change requires that a data upgrade script be run, consumers might no longer be able to synchronize their data model, or they might lose access to data.

### Data types

- **Changing an enumeration (enum) from extensible to non-extensible** – Consumers might have extensions to the enum.
- **Changing an enum from non-extensible to extensible** – Consumers might be using the enums in comparisons. For more details, see Write extensible enums.
- **Decreasing the decimal precision of an extended data type (EDT) of the real type** – Consumers might have dependencies on the ability to enter data that uses the precision.
- **Decreasing the size of an EDT of the string type** – Consumers might have dependencies on the size of the string.
- **Specializing the EDT by making it extend another EDT** – Consumers might have string length or decimal precision extensions to the EDT.
- **Changing the enum type of an EDT of the enum type when the enum is extensible** – Consumers might have extensions to the enum.

### Data model

- **Making a table obsolete and stopping information from being entered in the table** – Consumers might have a dependency on the table that information is entered in.
- **Making a table field obsolete and stopping information from being entered in the field** – Consumers might have a dependency on the field that information is entered in.
- **Renaming a field group** – Consumers might have extensions to field group, or code or metadata dependencies to it.
- **Changing constraints or indexes**
- **Making a table map obsolete and stopping the table map from being used**
- **Make a table map field obsolete**
- **Adding a table map field**
- **Removing a table map field mapping**
- **Making a data entity obsolete**
- **Making a data entity field obsolete**
Code changes

Class members

- **Deleting or renaming public or protected class-level members** – Consumers might be using these members in the extension classes.

- **Changing a class member from public or protected to protected or private** – Consumers might have queried or assigned values to the field.

Classes and interfaces

- **Adding an abstract method to a class** – Consumers might have created a derived type.

- **Adding final to a class** – Consumers might have created a derived type.

- **Adding a method to an interface** – Consumers might have implemented the interface on their own type.

- **Making a public class obsolete and stopping instantiation of the class** – Consumers might have overridden, wrapped, or subscribed to the instance methods.

Methods

- **Changing the access modifier from protected or public to another access modifier** – Consumers might have called, overridden, wrapped, or subscribed to the method.

- **Making a concrete public or protected method abstract** – Consumers might have subclasses to the class, and those subclasses might not have implemented the method, or they might even call `super`.

- **Renaming method parameters** – Consumers might have dependencies on parameters by name (via arguments or externally from C#, for example).

- **Adding, removing, or changing the type of a method parameter on a protected or public method** – Consumers might have called, overridden, wrapped, or subscribed to the method.

- **Adding or removing a default method parameter on a protected or public method** – Consumers might have wrapped or subscribed to the method.

- **Changing the return type of a method** – Consumers might have called, overridden, wrapped, or subscribed to the method.

- **Adding Hookable(false) to a protected or public method** – Consumers might have wrapped or subscribed to the method.

- **Adding Wrappable(false) to a protected or public method** – Consumers might have wrapped the method.

- **Removing Hookable(true) from a private or protected method** – Consumers might have subscribed to the method.

- **Removing Wrappable(true) from a private method** – Consumers might have wrapped the method.

- **Removing Replaceable(true) from a method** – Consumers might have conditionally wrapped the method.

- **Adding final to a protected or public method** – Consumers might have overridden, wrapped, or subscribed to the method.

- **Changing a method from instance to static or from static to instance** – Consumers might have called, overridden, wrapped, or subscribed to the method.

- **Making a method obsolete and stopping invocation of the method** – Consumers might have called, overridden, wrapped, or subscribed to the method.

- **Changing the responsibility of a method** – Consumers might have called, overridden, wrapped, or subscribed to the method.

- **Removing the reference to a method** – Consumers might have overridden, wrapped, or subscribed to the method, and they might expect their logic to run.

Delegates

- **Making any change in signature** – Consumers might have subscribed dynamically.
- **Removing the reference to a delegate** – Consumers might have subscribed, and they might expect their logic to run.

**Label changes**
- **Modifying or deleting a label** – Consumers might be using the label in the current context of the label text and the parameters that were passed, and so on. We recommend that, going forward, you add new labels in the event of a label change.

**Application element changes**
- **Removing any element** – Consumers might have a compile time dependency on the existence of the element.

**Metadata extensions**
- **Not following the naming guidelines for metadata or augmentation classes** – Consumers might have elements that have the same name.
The compatibility checker tool can detect metadata breaking changes against a specified baseline release or update. In this way, it helps ensure backward compatibility. Microsoft uses the tool to help ensure metadata compatibility.

The compatibility checker tool is available as one of the dev tools in Platform update 34. You can use it to ensure that your solutions are backward-compatible with earlier releases before you install or push updates to customers.

What the tool detects

The tool compares metadata of the current version with metadata of a baseline version. It detects and reports metadata changes that Microsoft has identified as breaking and added to the tool.

For a list of breaking changes that the tool detects, see the List of breaking changes detected by the tool section later in this topic.

NOTE

- The list in this topic doesn't include all the breaking changes that the tool can detect.
- The tool doesn't detect all breaking changes.

What the tool doesn't detect

The tool detects only breaking changes that can be identified by comparing data. For example, it doesn't detect the following breaking changes that often occur:

- The reference to a protected or public method is removed.
- The responsibility of a method is changed.

Using the tool

You can use the tool to detect metadata compatibility issues that a new version has against the version that it's replacing. Microsoft uses the tool to detect any breaking changes that a new monthly update has against the previous monthly update.

Usage

```
CompatibilityChecker.exe -BaselineDirectory=<Path to baseline metadata> -CurrentDirectory=<Path to current metadata> -ModuleName=<Module name> -OutputFile=<Output file path> -LogFile=<Log file path>
```

Example

```
CompatibilityChecker.exe -BaselineDirectory="\servername\archive\Build1\BaselineMetadata" -CurrentDirectory="E:\MyCode\retail\amd64\BaselineMetadata" -ModuleName="Directory" -OutputFile="E:\Logs\Directory\Diagnostics.xml" -LogFile="E:\Logs\Directory\Checkerlog.txt"
```

Description
The tool identifies breaking changes by comparing the current metadata with specified baseline metadata.

You must specify the following paths:

- **BaselineDirectory** – The path of the baseline metadata.
- **CurrentDirectory** – The path of the current (new) metadata.
- **OutputFile** – The path of the file that contains the list of breaking changes.

The following rules apply:

- You must compile the current metadata before you run the tool.
- The **OutputFile** file contains the list of breaking changes that the tool identifies.
- The **BaselineDirectory** directory must exist and must have the metadata for the specified module and its dependencies (if the module has any dependencies).
- The metadata paths for **BaselineDirectory** and **CurrentDirectory** should have metadata for **StaticMetadata**. This metadata should be present in a folder that is named **StaticMetadata** in the specified paths.
- You can suppress any breaking change that the tool identifies by adding an entry to the model’s ignore list. This file is present in the **AxIgnoreDiagnosticList** folder for the model.

---

**List of breaking changes detected by the tool**

---

**NOTE**

The tool identifies metadata compatibility changes as breaking changes only if they have been defined as breaking in the tool.

---

**Class members**

- **Changing the access modifier of protected or public class members** (including making a member read-only) – Consumers might have read from the field or assigned values to it.
- **Deleting or renaming public or protected class-level members** – Consumers might be using these members in some extension classes.

**Methods**

- **Changing the method signature of a protected or public method** – Wrappers and callers of the method will be broken.
- **Making a protected or public method obsolete** – Consumers might be wrapping or overriding the methods.

**Classes and interfaces**

- **Making a class final** – Consumers might have created a derived type.
- **Making a class abstract** – Consumers might be instantiating the class.
- **Adding an abstract method to a class** – Consumers might have created a derived type.
- **Adding a method to an interface** – Consumers might have implemented the interface on their own type.
- **Making a public class obsolete and stopping instantiation of the class** – Consumers might have overridden, wrapped, or subscribed to the instance methods.

**Delegates**

- **Any change in signature** – Consumers might have subscribed dynamically.

**Tables**

Any of the following changes will break table extensions and table references to tables and table fields:

- Deleting or renaming table fields, field groups, indexes, table mappings, or table relations
• Modifying these table properties: **Extends**, **SupportInheritance**, **TableType**, **SaveDataPerCompany.Yes**, or **SaveDataPerPartition**

• Modifying these table field properties: **ExtendedDataType**, **Scale**, or **String size**

• Modifying these table index properties: **AllowDuplicates.No** or **IndexType**

**Forms**

Any of the following changes will break form extensions that reference the controls or methods:

• Deleting or renaming form controls, form data sources, and form data source fields.

• All changes that are breaking for methods are also breaking for form methods.

**Enumerations (enums)**

• Modifying these properties: **IsExtensible** or **Value**

**Extended data types (EDTs)**

• Modifying these properties: **Extends**, **EnumType**, or **Scale**

**Entities**

• All changes that are breaking for tables are also breaking for entities.

• Renaming a public entity.

**Labels**

• **Modifying or deleting a label** – Consumers might be using the label in the current context of the label text and the parameters that were passed. We recommend that you add new labels instead of changing existing labels.

**Application elements**

• **Removing any element** – Consumers might have a compile-time dependency on the existence of the element.
This tutorial provides guidelines on how to take traces.

In this tutorial, you’ll take a tour of how to collect and download traces. The trace analysis tool works largely similar to the Microsoft Dynamics AX 2012 version, but it is not backward compatible and can’t be used to analyze AX 2012 traces. The trace parser tool can be found in the PerfSDK folder on your development deployments.

**Prerequisites**

This tutorial requires that you access the environment as an administrator on the instance. The administrator can also grant rights to other users to take a trace. In this way, you can trace scenarios that can’t be reproduced with administrative rights.

**Capture the trace**

1. Before you trace make sure your scenario is in a warm state, meaning that you have run the scenario you want to trace once before you take the trace. Being in a warm state prevents things like metadata loading and other possible warm-up tasks from being in the trace.

2. In the navigation bar, select Help, and then select Trace.

3. Name the trace that you are about to capture, and then select Start trace.

4. Perform actions that need to be analyzed, for example, opening Accounts payable > Vendors > All vendors.

5. When you are finished, select Stop trace. Then, you can select one of the following options (for this tutorial, select the second option):

   - **Download trace** – Store the captured trace on a local machine. You can analyze a downloaded trace with the desktop version of Trace Parser.

     **NOTE**
     
     If you download a trace it will not be available for later uploading.

   - **Upload trace** – Store the trace in the cloud for later downloading by, for example, the admin, it will be automatically deleted after 7 days and can also be deleted manually from the captured traces form.

     **NOTE**
     
     If your scenario takes more than 1-2 minutes it is better to try to take multiple smaller traces of 30 seconds each as the trace will likely get too big to be easily analyzed and there is a risk for losing data if the trace gets too big.

**Assign trace rights to user**

1. To give a user rights to capture a trace, go to System administration > Users > Users.

2. Select the user and assign the System tracing user role.
Remove the user role again once the user is done with tracing to avoid unwanted tracing.

Open captured trace

1. In the navigation bar, select Help, and then select Trace.
2. Select Captured traces.

**NOTE**
The captured traces button can only be seen by users with administrative rights.

3. Select the trace name to download to open and analyze it with the desktop version of trace parser or
4. Select the user name to get to the user options.
5. Delete the trace if you want. You might do delete the trace if you have downloaded it.

**NOTE**
The trace will be deleted after 7 days. For more information about the desktop version of trace parser, see Diagnose issues and analyze performance by using Trace parser.
Use the information in this topic to do single-user testing by using Visual Studio and the Performance software development kit (SDK) together with a performance test script that is generated by using Task recorder.

**IMPORTANT**

Visual Studio 2019 will be the last version of Visual Studio with web performance and load test features.

- If you are using the Visual Studio and Test Controller/Test Agent for on-premises load testing, Visual Studio 2019 will be the last version. You can continue using it until the end of the support cycle.
- For more information, see [Cloud-based load testing service end of life](#).

Use Task recorder to define and record an end-to-end business scenario

Before you run a single-user test, you must work with your business team to define your end-to-end scenarios and then use Task recorder to create a recording of the steps in each scenario. For more information about how to create a task recording, see [Task recorder resources](#). The scenarios that you should test depend on your customer's business requirements. In this topic, you will use the "Create and confirm a sales order" sample scenario.

1. Sign in as a Sales persona.

2. Turn on Task recorder, and create and confirm a sales order that includes the following information:
   - Customer account
   - Item number
   - Sales quantity
   - Site
   - Warehouse
   - Sales price

3. When you've finished, select **Save as developer recording** to download the XML file.

Configure a development environment


2. Unblock and unzip the files.

3. In the `dist` folder, rename the `.nupkg` files as `.zip` files, and then unzip them.

<table>
<thead>
<tr>
<th>ORIGINAL FILE NAME</th>
<th>NEW FILE NAME</th>
</tr>
</thead>
</table>
4. Under your PerfSDK folder, create a folder that is named Common\External\Selenium.

5. Copy the following files, and save them to the folder Common\External\Selenium that you created in the previous step:
   - IEDriverServer.exe from the unzipped IEDriverServer_Win32_3.13.0.zip file
   - WebDriver.dll and WebDriver.xml from the lib\net45 folder in the unzipped Selenium.WebDriver.StrongNamed.3.13.1.zip file

Generate a C# performance test from Task recorder

When you've finished recording the end-to-end scenario, you must generate a C# performance test script that is based the task recording.

1. In a development environment, open Microsoft Visual Studio as an admin.

2. From your PerfSDK folder, open the PerfSDKSample solution. In a tier-1 sandbox or a cloud-hosted environment, the PerfSDK folder is typically in <Service volumne>\PerfSDK\PerfSDKLocalDirectory.

3. Add a reference to the WebDriver.dll file in the Common\External\Selenium folder.
4. On the Dynamics 365 menu, point to Addins, and then select Create C# perf test from recording.

5. In the Import Task Recording dialog box, enter the following required details:

- **Recording path** – The file location of the developer recording of your end-to-end scenario.
- **Project path** – The location of the PerfSDKSample project. Typically, the path is `<Your_PerfSDK_Folder>\SampleProject\PerfSDKSample\PerfSDKSample.csproj`.
- **PerfSDK path** – The location of PerfSDK. Typically, the path is `<ServiceVolumeDrive>\PerfSDK\PerfSDKLocalDirectory`.

6. When you’ve finished, select Import. A new C# class is created under the Generated folder of your PerfSDKSample project.

7. Build the solution.

Run single-user testing by using Test Explorer in Visual Studio

1. Update the CloudEnvironment.config file of the PerfSDKSample project in the following ways, so that it reflects the configuration of your environment:

   - Verify that the **HostName** and **SOAPHostName** match your development environment.
   - Verify that the **UserName** for SelfMintingAdminUser matches the admin account of your
Tips and tricks

Use the following tips and tricks for single-user testing that uses Task recorder and the Performance SDK:

- Run your business end-to-end scenario first before you capture it by using Task recorder.
- When you record your scenario by using Task recorder, enter values manually instead of selecting them in drop-down lists.
- Replay your task recording to make sure that everything works as you expect.
- Restart Visual Studio if you don’t see your test case after the solution is built.

Troubleshooting

For information about single-user or multi-user testing that uses the Performance SDK, see Troubleshooting guide for single-user or multi-user testing with the Performance SDK.
This topic explains how to run multi-user testing by using Microsoft Visual Studio, the Performance software development kit (SDK), and the Task recorder test scripts.

**IMPORTANT**

Visual Studio 2019 will be the last version of Visual Studio that includes web performance and load testing features. If you're using the Visual Studio and Test Controller/Test Agent for on-premises load testing, Visual Studio 2019 will be the last version. You can continue to use it until the end of the support cycle. For more information, see Cloud-based load testing service end of life.

**Prerequisites**

Before you complete the steps in this topic, verify that the following prerequisites are met:

- You have **Visual Studio Enterprise edition** in a development environment. Enterprise edition is required to create load tests. If you're deploying your development box as a cloud-hosted environment through Microsoft Dynamics Lifecycle Services (LCS), be sure to select the appropriate Visual Studio version to deploy.
- The Visual Studio web performance and load testing tools are installed as described in Install the load testing component.
- You have a tier-2 or higher sandbox environment that has the same release (application version and platform update) as your development environment.
- You've configured your development environment by following the steps in Single-user testing with Task recorder and the Performance SDK.
- C# performance testing classes have been generated for your end-to-end (E2E) scenarios, and you can run a single-user test by following the steps in Single-user testing with Task recorder and the Performance SDK.

**Configure a development environment for multi-user testing**

The following configurations must be set up on the development machine that is used to locally host the testing controller and agent.

**NOTE**

For all Microsoft-managed sandboxes and sandboxes of the self-service type, Microsoft will generate the certificate for your environment and preconfigure it.

1. Create an environmental variable that is named **TestRoot**, and point it to the **PerfSDK** folder by running the following cmdlet in Windows PowerShell.

   ```
   [ENVIRONMENT]::SETENVIRONMENTVARIABLE("TESTROOT", "K:\PERFSDK\PERFSKDLALLOCALDIRECTORY", "USER")
   ```

   To verify the variable, run the following command in Windows PowerShell.
2. In LCS, open the Environment details page for your target sandbox environment.

On the Environment details page, the Maintain menu includes two new commands:

- Download RSAT certificate
- Regenerate RSAT certificate

3. Select Download RSAT certificate to retrieve the certificate bundle as a zip file.

4. You're warned that a clear-text password will be shown on screen. Select Yes to continue.

5. Copy the clear-text password, because you will need it later.

6. After the zip file is downloaded, unzip it. Inside, you should find a certificate (.cer) file and a personal information exchange (.pfx) file.

7. Double-tap (or double-click) the certificate (.cer) file to open it, and then select Install. Install this certificate on your local machine, and then browse to the Personal store. Repeat this process for the local machine location, and browse specifically to the Trusted Root Certification Authorities store.

8. Double-tap (or double-click) the personal information exchange (.pfx) file to open it, and then select Install. Install this certificate on your local machine, enter the password that you copied in step 5, and then browse to the Personal store. Repeat this process for the local machine location, enter the password that you copied in step 5, and browse specifically to the Trusted Root Certification Authorities store.

9. Double-tap (or double-click) the certificate file to open it. On the Details tab, scroll down until you find the Thumbprint section. Select Thumbprint, and copy the ID in the text box. Save this thumbprint to update the CloudEnvironment.config thumbprint for the Performance SDK.
NOTE
Microsoft will automatically rotate the certificate before it expires. At that time, you must download a new version of the certificate. For self-service environments, the certificate will be rotated every 90 days, during a downtime window that is closest to the expiry. Downtime windows include customer-initiated package deployment, and database movement operations that target the environment.

Prepare the PerfSDKSample solution for multi-user testing

Follow these steps to prepare the sample solution for performance testing. You can find the sample solution in the Performance SDK folder in your development environment. By default, the folder is at K:\PerfSDK\PerfSDKLocalDirectory.

1. Run the following cmdlets with elevated permissions to verify that the certificate that you installed earlier is correctly installed, and that the thumbprint that you saved earlier is in the Personal store on the local machine.

   ```powershell
   cd Cert:\LocalMachine\My
   Get-ChildItem | Where-Object { $_.Subject -like "CN=127.0.0.1" }
   ```

   The following illustration shows a sample result. Make sure that the thumbprint that you saved earlier is in the list.

   ![Thumbprint Illustration](image.png)

2. Update the CloudEnvironment.config configuration file in the Performance SDK folder to describe the targeted environment. As part of this update, follow these steps:

   a. Verify that the settings for HostName and SOAPHostName match your tier-2 or higher sandbox environment.
   b. Add the thumbprint that you saved earlier as the value for SelfSigningCertificateThumbprint. If the entry is missing from your configuration file, you can add it as shown in the illustration that follows.
   c. Update the setting of UserCount so that it matches the number of test users in your case.
   d. Update the setting of UserFormat so that it matches your naming convention for test users.
   e. In each AuthenticatorConfiguration element under the AuthenticatorConfigurationCollection element, replace
      ```powershell
      ```
      with
      ```powershell
      ```
   f. Comment out the AzureActiveDirectoryConfiguration and KeyVaultConfigurations elements.

   The result should resemble the following example.

   ```xml
   <CloudEnvironment config="..."/>
   ```

NOTE
If your Finance and Operations apps were deployed in 21Vianet, be sure to specify
```
```
3. Rename the `vsonline.testsettings` file `local.testsettings`.

4. Open the `local.testsettings` file in Visual Studio, and modify it by following these steps:
   
   a. In the **Test Settings** dialog box, on the **General** tab, in the **Test run Location** field group, select the **Run tests using local computer or a test controller** option.
   
   b. On the **Deployment** tab, select the **Enable deployment** checkbox, and then use the **Add Directory** button to add the `bin\debug` folder to the **Additional files and directories to deploy** field.
   
   ![Deployment Tab](image)

   c. On the **Hosts** tab, in the **Run tests in 32 bits or 64 bits process** field, select **Run test in 64 bits process on 64 bits machine**.
   
   d. Select **Apply**, and then close the **Test Settings** dialog box.
   
   e. Open your project configuration, and modify it by setting **Target Framework** to **.NET Framework 4.6.2**.

   ![Test Settings Dialog Box](image)

   **NOTE**

   Whenever you use the Microsoft Dynamics 365 Add-in to generate a C# performance test from a task recording, it will reload the project in Visual Studio instead of reopening the whole solution. Be sure to reload the solution before you run any load tests, to ensure that the test settings file is visible.

**Modify the performance test sources**

Follow these steps for each generated performance test in your solution.

1. Add the following statement at the top in the `using` directives section.

   ```csharp
   using MS.Dynamics.TestTools.UIHelpers.Core;
   ```
2. Modify the **TestSetup** method by replacing the whole body with the following lines.

```csharp
private DispatchedClient Client;
private UserContext _userContext;
private TimerProvider timerProvider;
[TestInitialize]
public void TestSetup()
{
    if (this.TestContext != null)
    {
        timerProvider = new TimerProvider(this.TestContext);
    }
    SetupData();
    Client = new DispatchedClientHelper().GetClient();
    Client.ForceEditMode = false;
    Client.Company = WellKnownCompanyID.USMF.ToString();
    Client.Open();
}
```

3. Modify the **TestCleanup** method so that it resembles the following example.

```csharp
public void TestCleanup()
{
    Client.Close();
    Client.Dispose();
    Client = Null;
}
```

4. Build your solution.

Add a test to the load test mix

Follow these steps to add a performance test to the test mix.

1. Open the **SampleLoadTest.loadtest** file, and find the **Test Mix** node.
2. Select and hold (or right-click) the **Test Mix** node, and then select **Edit Test Mix**.
3. In the **Edit Test Mix** dialog box, select **Add** to add your tests to the mix.
4. In the **Run Settings** node, modify the properties, and update the **Timing** fields for **Run Settings1**. These fields include **Warm-up Duration**, **Run Duration**, and **Cool-down Duration**.
5. In the Scenarios node, be sure to update the Load Pattern property, and set the Constant User Count parameter to the total number of users that you want to use to run the test.

Create test users

Test users must be added to the target environment. The naming pattern must match the pattern that is specified in the CloudEnvironment.config configuration file. You can either manually create the users in a Microsoft Dynamics 365 environment or use the MS.Dynamics.Performance.CreateUsers.exe console application in the Performance SDK folder.

If you manually create the users, make sure that the System Administrator security role is assigned to each user.

We recommend that you use the console application to create the users, because it reads the configuration files and calls the appropriate service endpoints.

Run multi-user testing by using a local test controller

1. In the Visual Studio project, open the SampleLoadTest.loadtest file, and select Run Load Test.

2. Review the test output.
Troubleshooting

For more information about single-user or multi-user testing that uses the Performance SDK, see Troubleshooting guide for single-user or multi-user testing with the Performance SDK.
Troubleshooting guide for testing with the Performance SDK

11/24/2021 • 12 minutes to read • Edit Online

No client was opened in the time-out period

This issue affects only single-user tests. When the test is running, a web client is opened, but a website is never loaded. Instead, there is an empty web client that has a white background. The following message appears at the top of the page, “This is the initial start page for the WebDriver server.” The test eventually times out and fails, and an error message is shown.

Error - No client was opened in the time-out period

Initialization method <Test class name>.TestSetup threw an exception. System.TimeoutException: System.TimeoutException: No client was opened in the timeout period.

Solution

See Multi-user testing using the Performance SDK. That topic explains how to create a correct certificate for this type of test. It also explains how to add the thumbprint of the certificate to the wif.config file.

Zoom factor

This issue affects only single-user tests.

Error - Zoom factor

Initialization method <Test class name>.TestSetup threw exception. System.InvalidOperationException: System.InvalidOperationException: Unexpected error launching Internet Explorer. Browser zoom level was set to 200%. It should be set to 100% (NoSuchDriver).

Solution - Zoom factor

In Internet Explorer, you can change the zoom factor to 100 percent by changing the following registry keys:

- Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Internet Explorer\Zoom\ResetZoomOnStartup = 0
- Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Internet Explorer\Zoom\ResetZoomOnStartup2 = 0
- Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Internet Explorer\Zoom\Zoomfactor = 80000

Depending on the version of the local machine that is used, before you start the Remote Desktop Protocol (RDP) session, you might have to select Change the size of text, apps and other items. This field is available in Display settings in Microsoft Windows.

If those steps don't work, change the size of your remote desktop before you start the RDP session, so that the default zoom level in Internet Explorer is 100 percent.

Certificate thumbprint errors

Error example - Certificate thumbprint errors

Solution - Certificate thumbprint errors

You might receive the error message for one of the following reasons:

- The certificate thumbprint that you copied into the CloudEnvironment.Config and wif.config files includes invisible Unicode characters. To determine whether the thumbprint contains invisible Unicode characters, paste it into a Unicode code converter, and see whether extra characters appear in the HTML/XML field. For example, you can use the Unicode converter that is available at https://r12a.github.io/apps/conversion/.

- The certificate wasn't installed on the Application Object Server (AOS) machine. To verify that the certificate can be found on the AOS machine, run the following Microsoft Windows PowerShell script.

```
cd Cert:\LocalMachine\My
Get-ChildItem | Where-Object { $_.Subject -like "CN=<name of your certificate>" }  
```

If the thumbprint doesn't appear in the Windows PowerShell console after you run the script, the certificate isn't installed. To fix the issue, copy and install a .cer file on all AOS machines.

- If this issue occurs when you run load tests, the setup scripts might not have installed the corresponding .pfx file correctly. Verify that the password that is specified in the CloudCtuFakeACSInstall.cmd file matches the password that was set when the certificate was created.

No endpoint is listening

This issue can occur when you run single-user or multi-user tests, or when you create users by using MS.Dynamics.Performance.CreateUsers.exe.

Error example - No endpoint is listening

The tests fail, or the user creation process fails, and the following error message is shown:

```
System.TypeInitializationException: The type initializer for 'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.UserManagement' threw an exception. --- > System.ServiceModel.EndpointNotFoundException: There was no endpoint listening at \<web address> that could accept the message. This is often caused by an incorrect address or SOAP action.
```

Solution - No endpoint is listening

This issue occurs when the host that is specified in the CloudEnvironment.Config file can't be accessed from the machine that is trying to run the tests or create users.
In the CloudEnvironment.Config file, review the values that are specified by the following keys:

- `<ExecutionConfigurations Key="HostName" Value="<web address of host>" />
- `<ExecutionConfigurations Key="SoapHostName" Value="<web address of SOAP>" />

The web addresses that are specified by these keys must be in the environment that you're testing. In a web browser on your developer machine, make sure that you can open the web address that is specified by the HostName key.

For online load tests, the environment that is specified by the HostName key in the CloudEnvironment.Config file must be publicly accessible from any machine. Therefore, if you must test a one-box environment, you won't be able to run the load test by using Microsoft Visual Studio Online, because the endpoint won't be accessible outside the one-box environment.

**Users can't be enumerated**

This issue can occur when you run multi-user tests, or when you create users by using MS.Dynamics.Performance.CreateUsers.exe.

**Error example - Users can't be enumerated**

```
  > System.InvalidOperationException: Could not enumerate AX users --->
  System.ServiceModel.FaultException`1[System.ComponentModel.Win32Exception]: Forbidden
```

**Solution - Users can't be enumerated**

Three scenarios can cause this error:

- The System Administrator role isn't assigned to the user who is specified as SelfMintingAdminUser in the CloudEnvironment.config file. To verify that you've specified the correct user, sign in to the endpoint, and view the user's roles.

- The user who is specified as SelfMintingAdminUser in the CloudEnvironment.config file has a provider other than https://sts.windows-ppe.net/ or https://sts.windows.net/. Sometimes, a company-specific domain is included in the Provider field for the admin user.

- If your Finance and Operations apps were deployed in 21Vianet, make sure that you have specified NetworkDomain="https://sts.chinacloudapi.cn/" in SelfMintingSysUser and SelfMintingAdminUser.
To work around this issue, create a user who has any name and email address. Assign the **System Administrator** role to the new user. You don’t have to link the user to a real Microsoft Azure Active Directory (Azure AD) user. Specify this new admin user as **SelfMintingAdminUser** in the CloudEnvironment.config file.

**The HTTP request was forbidden with client authentication scheme 'Anonymous'**

**Error example - The HTTP request was forbidden**

Initializes method &lt;Test class name&gt;.TestSetup threw exception.

```csharp
```

**Solution - The HTTP request was forbidden**

Three known scenarios can cause this error:

- The test users are created by running MS.Dynamics.Performance.CreateUsers.exe without any arguments. For example, if the CreateUsers script is run without any arguments, the email addresses of test users that are created won’t be correctly formatted. If these users are used to run the tests, the tests will generate the forbidden request error. You can verify that this scenario is causing the error by viewing the users. The incorrect email addresses of the test users will resemble the email addresses in the following illustration.

<table>
<thead>
<tr>
<th>User ID</th>
<th>User name</th>
<th>Email Address</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D08C</td>
<td>$D08C</td>
<td><a href="mailto:TestBadFormat_5@TAEOfficial.ccctp.net">TestBadFormat_5@TAEOfficial.ccctp.net</a> Admin (<a href="mailto:admin@as7partner.ccctp.net">admin@as7partner.ccctp.net</a>)</td>
<td>USMF</td>
</tr>
<tr>
<td>$4ED9</td>
<td>$4ED9</td>
<td><a href="mailto:TestBadFormat_10@TAEOfficial.ccctp.net">TestBadFormat_10@TAEOfficial.ccctp.net</a> Admin (<a href="mailto:admin@as7partner.ccctp.net">admin@as7partner.ccctp.net</a>)</td>
<td>USMF</td>
</tr>
<tr>
<td>$6CB7</td>
<td>$6CB7</td>
<td><a href="mailto:TestBadFormat_2@TAEOfficial.ccctp.net">TestBadFormat_2@TAEOfficial.ccctp.net</a> Admin (<a href="mailto:admin@as7partner.ccctp.net">admin@as7partner.ccctp.net</a>)</td>
<td>USMF</td>
</tr>
<tr>
<td>$8CE0</td>
<td>$8CE0</td>
<td><a href="mailto:TestBadFormat_9@TAEOfficial.ccctp.net">TestBadFormat_9@TAEOfficial.ccctp.net</a> Admin (<a href="mailto:admin@as7partner.ccctp.net">admin@as7partner.ccctp.net</a>)</td>
<td>USMF</td>
</tr>
<tr>
<td>$807A</td>
<td>$807A</td>
<td><a href="mailto:TestBadFormat_1@TAEOfficial.ccctp.net">TestBadFormat_1@TAEOfficial.ccctp.net</a> Admin (<a href="mailto:admin@as7partner.ccctp.net">admin@as7partner.ccctp.net</a>)</td>
<td>USMF</td>
</tr>
<tr>
<td>$C600</td>
<td>$C600</td>
<td><a href="mailto:TestBadFormat_8@TAEOfficial.ccctp.net">TestBadFormat_8@TAEOfficial.ccctp.net</a> Admin (<a href="mailto:admin@as7partner.ccctp.net">admin@as7partner.ccctp.net</a>)</td>
<td>USMF</td>
</tr>
<tr>
<td>$D148B</td>
<td>$D148B</td>
<td><a href="mailto:TestBadFormat_16@TAEOfficial.ccctp.net">TestBadFormat_16@TAEOfficial.ccctp.net</a> Admin (<a href="mailto:admin@as7partner.ccctp.net">admin@as7partner.ccctp.net</a>)</td>
<td>USMF</td>
</tr>
<tr>
<td>$EADF</td>
<td>$EADF</td>
<td><a href="mailto:TestBadFormat_15@TAEOfficial.ccctp.net">TestBadFormat_15@TAEOfficial.ccctp.net</a> Admin (<a href="mailto:admin@as7partner.ccctp.net">admin@as7partner.ccctp.net</a>)</td>
<td>USMF</td>
</tr>
<tr>
<td>$EDE4</td>
<td>$EDE4</td>
<td><a href="mailto:TestBadFormat_7@TAEOfficial.ccctp.net">TestBadFormat_7@TAEOfficial.ccctp.net</a> Admin (<a href="mailto:admin@as7partner.ccctp.net">admin@as7partner.ccctp.net</a>)</td>
<td>USMF</td>
</tr>
<tr>
<td>$FA4A</td>
<td>$FA4A</td>
<td><a href="mailto:TestBadFormat_6@TAEOfficial.ccctp.net">TestBadFormat_6@TAEOfficial.ccctp.net</a> Admin (<a href="mailto:admin@as7partner.ccctp.net">admin@as7partner.ccctp.net</a>)</td>
<td>USMF</td>
</tr>
</tbody>
</table>

To resolve the issue, delete the test users who have incorrectly formatted email addresses. Rerun the CreateUsers script, and specify the user count and company.

- The number of users that you specify in the **UserCount** field in the CloudEnvironment.Config file exceeds the number of test users that you created by using MS.Dynamics.Performance.CreateUsers.exe. Make sure that you created at least as many test users as you request in the CloudEnvironment.Config file.
At least one security token in the message could not be validated

This issue can occur when you run multi-user tests, when you create users by using MS.Dynamics.Performance.CreateUsers.exe, when the AOS machine differs from the developer machine.

Error example - At least one security token in the message could not be validated

This issue occurs when the AOS endpoint can’t validate the thumbprint of the certificate that you created. There are two possible causes:

- The certificate wasn’t installed on the AOS machine. To fix the issue, copy a .cer file to the AOS machine, and install it.
- The thumbprint of the certificate wasn’t added to the wif.config file on the AOS machine. To fix the issue, add the certificate to the wif.config file. Be sure to restart Microsoft Internet Information Services (IIS) after you change the wif.config file.

Solution - At least one security token in the message could not be validated

This issue usually occurs when you run load tests.

Error example - MS.Dynamics.Test.Team.Foundation.WebClient.InteractionService.dll.config is missing from the deployment items

This issue usually occurs when you run load tests.

This issue occurs when the system can’t find the `MS.Dynamics.Test.Team.Foundation.WebClient.InteractionService.dll.config` file when the load tests are run, because the file wasn’t added as a deployment item. Verify that the `MS.Dynamics.Test.Team.Foundation.WebClient.InteractionService.dll.config` file is in the Out folder for the test run:

```
<solution path>\TestResults\<your test run>\Out
```

If the file is missing, add it to the deployment items in the test settings.

There are two files that have very similar names. The name of one file ends in *.dll, and the name of the other file ends in *.dll.config. The *.dll.config file must be in the deployment items in the test settings.

CloudEnvironment.Config is missing from the deployment items

This issue usually occurs only when you run load tests.

Error example - `CloudEnvironment.Config is missing`

```
```

Solution - `CloudEnvironment.Config is missing`

This issue occurs when the CloudEnvironment.Config file isn’t present when the tests are run. It typically occurs when you run load tests and the CloudEnvironment.Config file wasn’t added as a deployment item. Verify that the CloudEnvironment.Config file is in the Out folder for the test run:

```
<solution path>\TestResults\<your test run>\Out
```

If the file is missing, add it to the deployment items in the test settings.
InteractiveClientId was not specified in the settings

Error example - InteractiveClientId was not specified in the settings

The type initializer for


--->
Microsoft.CE.VaultSDK.SecretProviderException: InteractiveClientId was not specified in settings.

Solution - InteractiveClientId was not specified in the settings

This issue occurs when the SelfSigningCertificateThumbprint field is left blank in the CloudEnvironment.Config file. In the CloudEnvironment.Config file, find the following line, and paste in the thumbprint of the certificate that you created and installed.

\<ExecutionConfigurations Key="SelfSigningCertificateThumbprint" Value="" />

The remote host forcibly closed an existing connection

Error example - The remote host forcibly closed an existing connection

System.TypeInitializationException: System.TypeInitializationException: The type initializer for


--->
System.ServiceModel.CommunicationException: An error occurred while making the HTTP request to \\Host name\Services\AxUserManagement\Service.svc/ws2007FedHttp. This could be due to the fact that the server certificate is not configured properly with HTTP.SYS in the HTTPS case. This could also be caused by a mismatch of the security binding between the client and the server.

--->
System.Net.WebException: The underlying connection was closed: An unexpected error occurred on a send.

--->
System.IO.IOException: Unable to read data from the transport connection: An existing connection was forcibly closed by the remote
host. ---> System.Net.Sockets.SocketException: An existing connection was forcibly closed by the remote host.

Solution - The remote host forcibly closed an existing connection
Run the following Windows PowerShell script on the development machine.

```powershell
Set-ItemProperty HKLM:\SOFTWARE\Microsoft\.NETFramework\v4.0.30319 -Name SchUseStrongCrypto -Value 1 -Type dword -Force -Confirm:$false
if ((Test-Path HKLM:\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\v4.0.30319))
{
    Set-ItemProperty HKLM:\SOFTWARE\Wow6432Node\Microsoft\NETFramework\v4.0.30319 -Name SchUseStrongCrypto -Value 1 -Type dword -Force -Confirm:$false
}
```

Service w3svc was not found on computer

This error occurs only when you run load tests by using Visual Studio Online.

Error example - Service w3svc was not found on computer

Test method
System.ComponentModel.Win32Exception: The specified service does not exist as an installed service.

Solution - Service w3svc was not found on computer

A hotfix is available that resolves this issue. The Microsoft Knowledge Base (KB) number is 4095640.

The file IEDriverServer.exe does not exist

This issue affects only single-user tests.

Error example - The file IEDriverServer.exe does not exist

The file K:\perfSDK\PerfSDKLocalDirectory\SampleProject\TestResults\Admin501201994c_devae648d1909-1 2018-06-25 03_40_51\Out\Common\External\Selenium\IEDriverServer.exe does not exist. The driver can be downloaded at https://selenium-release.storage.googleapis.com/index.html.

Solution - The file IEDriverServer.exe does not exist

Copy the Common\External\Selenium folder under <Your_PerfSDK_Folder> to the <Your_PerfSDK_Folder>\SampleProject\PerfSDKSample\bin\Debug folder.

Failed finding the certificate for minting tokens by thumbprint: <your certificate thumbprint>

Error example - Failed finding the certificate for minting tokens by thumbprint
Solution - Failed finding the certificate for minting tokens by thumbprint
Make sure that you install the generated certificate on each AOS machine in your sandbox environment.

The action you are trying to perform requires a connection to Visual Studio Team Services
This issue affects only multi-user tests.

Error example - The action you are trying to perform requires a connection to Visual Studio Team Services

Solution - The action you are trying to perform requires a connection to Visual Studio Team Services
When you connect to Azure DevOps, use the old URI format (<Azure_DevOps_Account>.visualstudio.com) instead of dev.azure.com/<Azure_DevOps_Account>. Additionally, open Azure DevOps by using the old URI, and then select Open in Visual Studio.

Could not load file or assembly 'aoskernel.dll' or one of its dependencies
This error affects only multi-user tests.

Error example - Could not load file or assembly 'aoskernel.dll'

Solution - Could not load file or assembly 'aoskernel.dll'
Make sure that you're using Open Database Connectivity (ODBC) Driver 17 in an environment that has Platform update 20 or later.

AzureActiveDirectoryConfiguration node is missing in CloudEnvironment.config

Error example - AzureActiveDirectoryConfiguration node is missing
Solution - AzureActiveDirectoryConfiguration node is missing


Multiple warning messages before and after multi-user testing that uses Azure DevOps

Error example - Multiple warning messages before and after multi-user testing

There is no impact, and the messages can be ignored.

The type or namespace name 'xxxx' could not be found (are you missing a using directive or an assembly reference?)

Error example - The type or namespace name 'xxxx' could not be found

The type or namespace name 'InventTransferOrders' could not be found (are you missing a using directive or an assembly reference?)

Solution - The type or namespace name 'xxxx' could not be found

The sample solution shipped with the perfSDK was previously prepared and wasn't updated after the packages split. To resolve the issue, add the assembly MS.Dynamics.TestTools.DirectoryProxyLibrary.dll under <Service volume>:\PerfSDK\PerfSDKLocalDirectory as a reference.

Assembly was built against the ".NETFramework,Version=v4.6" framework

Error example - Assembly was built against the ".NETFramework,Version=v4.6" framework
The primary reference “MS.Dynamics.TestTools.ApplicationSuiteProxyLibrary” could not be resolved because it has an indirect dependency on the assembly “MS.Dynamics.TestTools.DirectoryProxyLibrary, Version=7.0.0.0, Culture=neutral, PublicKeyToken=a7cf325ee2c8a9ff” which was built against the “.NETFramework,Version=v4.6” framework. This is a higher version than the currently targeted framework “.NETFramework,Version=v4.5”.

**Solution- Assembly was built against the ".NETFramework,Version=v4.6" framework**

Change the **Target framework** property in the properties window of PerfSDKSample to .Net Framework 4.6.
This topic explains how to use the Performance software development kit (SDK) to do multiuser load testing in an on-premises environment.

**Prerequisites**

- An on-premises environment that has volume data
- A development environment that has the following characteristics:
  - Microsoft Visual Studio Enterprise or a later version is installed.
  - The Performance SDK is installed. (The SDK will likely be in K:\PerfSDK\PerfSDKLocalDirectory. However, depending on your environment, it might be in another location, such as C:\PerfSDK.)
  - The on-premises environment can be accessed in a web browser. (The development virtual machine [VM] might be in the same domain as the on-premises environment, or the on-premises environment might have a publicly registered domain name.)

**Create a single-user C# test from an XML recording**

1. Use Task recorder to create a recording of the scenario that you want to test.

   **IMPORTANT**
   
   Your task recording must start on the default dashboard page. Otherwise, the test won’t be able to run.

2. Start Microsoft Visual Studio as an administrator, and build the PerfSDKSample project. This project is in the PerfSDK folder. If you’ve already built the project, skip this step.

3. Select Dynamics 365 > Addins > Create C# perf test from recording.

4. In the Import Task Recording dialog box, enter the required details, and then select Import.
Run a single-user test by using the Performance SDK

Prepare the development environment

Follow these steps in the development environment.

1. In Control Panel in Microsoft Windows, select System and Security > System > Advanced System Settings. Verify that the TestRoot environment variable is set to the path of the PerfSDK folder.

2. Download the selenium-dotnet-strongnamed-2.42.0.zip and IEDriverServer_Win32_2.42.0.zip files from https://selenium-release.storage.googleapis.com/index.html?path=2.42/, and extract the files.

3. Copy the dynamic-link libraries (DLLs) from the selenium-dotnet-strongnamed-2.42.0.zip\net40 folder to the PerfSDK\Common\External\Selenium folder. Also copy the IEDriverServer.exe from the IEDriverServer_Win32_2.42.0.zip to the PerfSDK\Common\External\Selenium folder.
4. Generate a certificate to use for authentication for the tests. To generate a certificate file, open a Command Prompt window as an administrator, and run the following commands. When you're prompted for a private key password, select None.

```
"C:\Program Files (x86)\Windows Kits\8.1\bin\x64\makecert" -n "CN=127.0.0.1" -ss Root -sr LocalMachine -a sha256 -len 2048 -cy end -r -eku 1.3.6.1.5.5.7.3.1 -sv c:\temp\authcert.pvk c:\temp\authcert.cer
"c:\Program Files (x86)\Windows Kits\8.1\bin\x64\pvk2pfx" -pvk c:\temp\authCert.pvk -spc c:\temp\authcert.cer -pfx c:\temp\authcert.pfx
```

Note the following elements in these commands:

- `-n "CN=127.0.0.1"` gives a human-readable name to the certificate. The name of this certificate must be 127.0.0.1. Otherwise, the single-user tests won't be able to run.
- `-eku 1.3.6.1.5.5.7.3.1` gives the purpose of the certificate. It indicates that the certificate can be used as a Secure Sockets Layer (SSL) server certificate.

After the script has finished running, you should see the following files in C:\Temp:

- authcert.pfx
- authcert.cer
- authcert.pvk

5. Install the authcert.pfx certificate file. When you install the file, make sure that you select Local Machine.

6. Copy the authcert.pfx file to the PerfSDK folder.

7. Open a Microsoft Windows PowerShell window as an administrator, and run the following commands to get the thumbprint of the installed certificate.

```
cd Cert:\LocalMachine\My
Get-ChildItem | Where-Object { $_.Subject -like "CN=127.0.0.1" }
```

8. In Visual Studio, open the PerfSDKSample project that is in the PerfSDK folder.

9. In the Visual Studio project, add a reference to the WebDriver.dll file in the PerfSDK\Common\External\Selenium folder.
10. Open the CloudEnvironment.Config file, and replace the contents with the following template.

```xml
<?xml version="1.0" encoding="utf-8"?>
<EnvironmentalConfigSettings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<EnvironmentalConfigSettingsCollection>
<EnvironmentalConfigSetting ConfigName="DEVFABRIC">
  <!-- NOTE: the HostName value needs to be specified -->
  <ExecutionConfigurations Key="HostName" Value="[yourD365FOdomain]/namespaces/AXSF" />
  <ExecutionConfigurations Key="SoapHostName" Value="[yourD365FOdomain]/namespaces/AXSF" />
  <ExecutionConfigurations Key="SelfSigningCertificateThumbprint" Value="[ThumbprintFromPowerShell]" />
  <ExecutionConfigurations Key="AdminAuthenticatorConfigurationId" Value="SelfMintingAdminUser" />
  <ExecutionConfigurations Key="DefaultBrowser" Value="InternetExplorer" />
  <ExecutionConfigurations Key="FederationRealm" Value="spn:00000015-0000-0000-c000-00000000000" />
  <ExecutionConfigurationsNodes ConfigurationName="SVC">
    <ConfigurationSpecificDetails Key="AppConfig" Value="DEVFABRIC.Config" />
  </ExecutionConfigurationsNodes>
  <ExecutionConfigurationsNodes ConfigurationName="PRF">
    <ConfigurationSpecificDetails Key="IsAdfs" Value="True" />
    <ConfigurationSpecificDetails Key="UserCount" Value="2" />
    <ConfigurationSpecificDetails Key="UserFormat" Value="[AdminUserEmail]" />
    <ConfigurationSpecificDetails Key="UserPassword" Value="[AdminUserPassword]" />
    <ConfigurationSpecificDetails Key="UserRole" Value="-SYSADMIN-" />
    <ConfigurationSpecificDetails Key="ThinkTime" Value="0" />
    <ConfigurationSpecificDetails Key="Company" Value="USMF" />
  </ExecutionConfigurationsNodes>
</EnvironmentalConfigSetting>
</EnvironmentalConfigSettingsCollection>
<AuthenticatorConfigurationCollection>
<AuthenticatorConfiguration Id="SelfMintingRunnerUser"
  <Credentials IsFromKeyVault="false" Username="daxrunneruser@daxmdsrunner.com"
    NetworkDomain="urn:Microsoft:Dynamics:Cloud:DaxRunner" />
</AuthenticatorConfiguration>
<AuthenticatorConfiguration Id="SelfMintingSysUser"
  <Credentials IsFromKeyVault="false" Username="testuser@microsoft.com" />
</AuthenticatorConfiguration>
<AuthenticatorConfiguration Id="SelfMintingAdminUser"
  <!-- NOTE: admin username needs to be specified -->
  <Credentials IsFromKeyVault="false" Username="[AdminUserEmail]"
    NetworkDomain="[AdfsUrl]" />
</AuthenticatorConfiguration>
</AuthenticatorConfigurationCollection>
</EnvironmentalConfigSettings
```

11. In the CloudEnvironment.Config file, specify values for the following keys. These values replace the placeholder values in square brackets in the template.

- **HostName** – Specify the URL that is used to access your on-premises environment. The URL should be `[yourD365FOdomain]/namespaces/AXSF`.

- **SoapHostName** – Specify the same URL that you specified for **HostName**.

- **SelfSigningCertificateThumbprint** – Specify the thumbprint that you retrieved from Windows PowerShell in step 7.

- **UserFormat** – Specify the email address of a user who has the System Administrator role in your
on-premises environment. The user must be an Active Directory Federation Services (AD FS) user.

- **UserPassword** – Specify the password of the user whose email address you specified for **UserFormat**.
- **Username** – Specify the same email address that you specified for **UserFormat**.
- **NetworkDomain** – Specify the URL of the AD FS identity provider. You can find this value by running the following SQL query against the **AXDB** database of your on-premises deployment.

```sql
select NETWORKDOMAIN, NETWORKALIAS from USERINFO where NETWORKALIAS='[AdminUserEmail]'
```

**Prepare the on-premises environment**

Follow these steps on each Application Object Server (AOS) VM in the on-premises deployment.

1. Copy the **authcert.cer** file that you created in the *Prepare the development environment* section of this topic to the AOS VM.
2. Install the **authcert.cer** certificate file. When you install the certificate, make sure that you select **Local Machine**. Also make sure that you put the certificate in the **Trusted Root Certification Authorities** store.
3. Open the **wif.config** file in a text editor. The path of the file will resemble

C:\ProgramData\SF\AOS1\Fabric\work\Applications\AXSFType_App19\AXSF.Code.1.0.20180 717001108.

**NOTE**

In the file path, the AOS number (**AOS1** in this example) will vary, depending on the AOS node that you're on. Additionally, the **ProgramData** folder is a hidden folder. Therefore, to see the folder in File Explorer, you must enable hidden items.

4. In the **wif.config** file, find the authority that is named [https://fakeacs.accesscontrol.windows.net](https://fakeacs.accesscontrol.windows.net]. In the list of thumbprints for this authority, add the thumbprint of the certificate that you created in the *Prepare the development environment* section. In the following example, the fourth thumbprint has been added to the [https://fakeacs.accesscontrol.windows.net](https://fakeacs.accesscontrol.windows.net) authority.

```xml
<authority name="https://fakeacs.accesscontrol.windows.net/">  
  <keys>  
    <add thumbprint="956780F32F4B312FE44A5CE88A9CAAA13B7FB8B6" />  
    <add thumbprint="B8F659E2FDD2A6811CD72BE753FF7ACB64351AC1" />  
    <add thumbprint="B51C394E6E957BD8C54AE0A9927DBB66C6E0F84B" />  
    <add thumbprint="F9112DE34DE65CBE3961EBDC7FB1F8525DED" />  
  </keys>  
  <validIssuers>  
    <add name="https://fakeacs.accesscontrol.windows.net/" />  
  </validIssuers>  
</authority>
```

5. Open the **AXService.exe.config** file in a text editor. You can find this file in the same directory as the **wif.config** file.

6. Search the **AXService.exe.config** file for "**Aos.AosRole**". Replace the line that contains "**Aos.AosRole**" with the following line.
7. In Service Fabric Explorer, find the **Code** package for the AOS node, select the ellipse button (...), and then select **Restart** to restart the application.

Run the single-user test

1. In the **PerfSDKSample** project, find the **PurchaseReq.cs** file. This file is a sample single-user test. In the file, comment out the following lines.

```csharp
if (this.TestContext !=null)
{
    timerProvider = new TimerProvider(this.TestContext);
}
```
2. Select Test > Test settings, set the Default processor architecture field to x64, and then build the solution.

3. Select Test > Windows > Test Explorer to view the list of tests.

NOTE
Sometimes, Visual Studio might not update the list of tests after you create a test script from a task recording. In this case, restart Visual Studio, and then reopen Test Explorer.

4. Run the sample single-user test by right-clicking CreatePurchReq. Alternatively, you can run the test that you created from your task recording. When you run the test, Internet Explorer should be started, and it should replay the scenario that you recorded.

Run a multiuser load test by using the Performance SDK

Create a multiuser test from a single-user test
After you create a single-user test by using the information earlier in this topic, you can convert it to a multiuser test. Add MS.Dynamics.TestTools.UIHelpers.Core; to your test script, and find the following line in the TestSetup method.

```
Client = DispatchedClient.DefaultInstance;
```

Replace that line with the following lines.

```
DispatchedClientHelper helper = new DispatchedClientHelper();
Client = helper.GetClient();
```

The test script that was generated by the Task Importer might contain a line that resembles the following line.

```
UserContextRole _context = new UserContextRole(UserManagement.AdminUser);
```

Remove this line from any tests that will be run as load tests. This code is required only for single-user tests and has a negative effect on the performance of load tests.

Make sure that the values that you entered when you made the task recording are randomized.

Run the multiuser load test
1. In the Visual Studio editor, open the ProcureToPay.cs file, and append the following lines in the TestSetup method.
var testroot = System.Environment.GetEnvironmentVariable("DeploymentDir");
if (string.IsNullOrEmpty(testroot))
{
    testroot = System.IO.Directory.GetCurrentDirectory();
}
Environment.SetEnvironmentVariable("testroot", testroot);

2. Download the installer (.msi) file for Microsoft ODBC Driver 13 for SQL Server from
https://www.microsoft.com/download/details.aspx?id=50420. (Select the 64-bit version of the .msi file.)
Put the file in the Visual Studio Online folder in the PerfSDK directory.

3. Modify the contents of the setup.cmd file in the Visual Studio Online folder so that they match the
following code.

```powershell
setx testroot "%DeploymentDirectory%"
ECHO Installing D365 prerequisites
ECHO MSIEXEC /a %DeploymentDirectory%\msodbcsql /passive /norestart IACCEPTMSODBCSQLLICENSETERMS=YES
ECHO MSIEXEC /a %DeploymentDirectory%\msodbcsql /passive /norestart IACCEPTMSODBCSQLLICENSETERMS=YES
ECHO %windir%\sysnative\windowspowershell\v1.0\powershell.exe -File %DeploymentDirectory%\install-wif.psl
Md %DeploymentDirectory%\Common\Team\Foundation\Performance\Framework
%DeploymentDirectory%\CloudCtuFakeACSInstall.cmd %DeploymentDirectory%\authcert.pfx
```

4. Modify the contents of the CloudCtuFakeACSInstall.cmd file so that the Import command has an
empty string instead of 'password'. The third line of the script should resemble the following line.

```powershell
set MyStoreInstallCmd= .... $pfxcert.Import('%TestCertPath%', '', 'Exportable,PersistKeySet')....
```

5. In your solution files, double-click the vsonline.testsettings file to modify the test settings.

6. In the Test Settings dialog box, on the General tab, set the Test run location field to Run tests using
local computer or a test controller.

7. On the Deployment tab, use the following settings:
   - Select the Enable deployment check box.
   - In the Additional files and directories to deploy field, make sure that the following files and
directories are listed:
     - <Solution Directory>\PerfSDKSample\bin\Debug\n     - C:\PerfSDK\CloudEnvironment.Config
     - C:\PerfSDK\authcert.pfx
     - C:\PerfSDK\Visual Studio Online\```
8. On the Setup and Cleanup Scripts tab, select the setup.cmd file that is in the Visual Studio Online folder in the PerfSDK directory.

9. On the Hosts tab, select Run tests in 64 bit process on 64 bit machine.

10. To run the test, open the SampleLoadTest.loadtest file, and select Run Load Test.

When the test has finished running, you should see a summary that shows transaction results. Here is an example.

<table>
<thead>
<tr>
<th>Test Results</th>
<th>Scenario</th>
<th>Total Tests</th>
<th>Failed Tests (% of total)</th>
<th>Avg. Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProcureToPaymentTrend</td>
<td>Scenario1</td>
<td>10</td>
<td>10 (100)</td>
<td>56.6</td>
</tr>
</tbody>
</table>

11. To view various indicators for the test controller and test scenario, you can switch to the Graphs view.
Troubleshooting

Zoom factor

This issue affects only single-user tests.

Error example

Initialization method <Test class name>.TestSetup threw exception. System.InvalidOperationException: System.InvalidOperationException: Unexpected error launching Internet Explorer. Browser zoom level was set to 200%. It should be set to 100% (NoSuchDriver).

Solution

In Internet Explorer, you can change the zoom factor to 100 percent by changing the following registry keys:

- `Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Internet Explorer\Zoom\ResetZoomOnStartup = 0`
- `Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Internet Explorer\Zoom\ResetZoomOnStartup2 = 0`
- `Computer\HKEY_CURRENT_USER\SOFTWARE\Microsoft\Internet Explorer\Zoom\Zoomfactor = 80000`

Depending on the version of the local machine that is used, before you start the Remote Desktop Protocol (RDP) session, you might have to select Change the size of text, apps and other items. This field is available in Display settings in Windows.

If those steps don’t work, try to change the size of your remote desktop before you start the RDP session, so that the default zoom level in Internet Explorer is 100 percent.

Certificate thumbprint errors

Error example


Solution

You might receive the error message for several reasons:
The certificate thumbprint that you copied into the CloudEnvironment.Config and wif.config files includes invisible Unicode characters. To determine whether the thumbprint contains invisible Unicode characters, paste it into a Unicode code converter, and see whether extra characters appear in the HTML/XML field. For example, you can use the Unicode converter that is available at https://12a.github.io/apps/conversion/.

The certificate wasn’t installed correctly on the AOS machine. To verify that the certificate can be found on the AOS machine, run the following Windows PowerShell script.

```powershell
cd Cert:\LocalMachine\My
Get-ChildItem | Where-Object { $_.Subject -like "CN=127.0.0.1" }
```

If the thumbprint doesn’t appear in the Windows PowerShell console after you run the script, the certificate can’t be found. To fix the issue, copy and install the .cer file that you created earlier in this topic to the AOS machine.

If this issue occurs when you run load tests, the setup scripts might not have installed the corresponding .pfx file correctly. Verify that the password that is specified in the CloudCtuFakeACSInstall.cmd file matches the password that was set when the certificate was created.

The tests process fails, and the following error message is shown.

```
System.TypeInitializationException: The type initializer for 'MS.Dynamics.TestTools.CloudCommonTestUtilities.Authentication.UserManagement' threw an exception.  ---> System.ServiceModel.EndpointNotFoundException: There was no endpoint listening at <web address> that could accept the message. This is often caused by an incorrect address or SOAP action.
```

Solution
This issue occurs when the host that is specified in the CloudEnvironment.Config file can’t be accessed from the machine that is trying to run the tests or create users.

In the CloudEnvironment.Config file, review the values that are specified for the following keys:

- `<ExecutionConfigurations Key="HostName" Value="<web address of host>" />`
- `<ExecutionConfigurations Key="SoapHostName" Value="<web address of SOAP>" />

The web addresses that are specified by these keys must be the environment that you’re testing. In a web browser on your developer machine, make sure that you can open the web address that is specified for the HostName key.

Users can’t be enumerated
This issue can occur when you run multiuser tests, or when you create users by using MS.Dynamics.Performance.CreateUsers.exe.
Error example

System.InvalidOperationException: Could not enumerate AX users --->
System.ServiceModel.FaultException'1[System.ComponentModel.Win32Exception]: Forbidden

Solution

Two scenarios can cause this error:

- The user who is specified as **SelfMintingAdminUser** in the CloudEnvironment.Config file must have the System Administrator role. This issue occurs when the System Administrator role isn’t assigned to the user who is specified as **SelfMintingAdminUser**. To verify that you’ve specified the correct user, you can sign in to the endpoint and view the user’s roles.

- An incorrect **NetworkDomain** value was specified for the user who is specified as **SelfMintingAdminUser** in the CloudEnvironment.Config file. You can find the correct value by running the following SQL query against the **AXDB** database of your on-premises deployment.

```sql
select NETWORKDOMAIN, NETWORKALIAS from USERINFO where NETWORKALIAS='[AdminUserEmail]'
```

**At least one security token in the message could not be validated**

This issue can occur when you run multiuser tests, or when you create users by using MS.Dynamics.Performance.CreateUsers.exe. It tends to occur when the AOS machine differs from the developer machine.

Error example

System.ServiceModel.ServiceModel.Security.MessageSecurityException: An unsecured or incorrectly secured fault was received from the other party. See the inner FaultException for the fault code and detail. --->
System.ServiceModel.FaultException: At least one security token in the message could not be validated.

Solution

This issue occurs when the AOS endpoint can't validate the thumbprint of the certificate that you created. There are three possible causes:

- In the CloudEnvironment.Config file, either a value isn’t specified for the **IsAdfs** key, or the value is set to **False**. Make sure that the value for the **IsAdfs** key is set to **True**.

- The certificate wasn't installed on the AOS machine. To fix the issue, copy the .cer file that you created earlier
in this topic to the AOS machine, and install the certificate.

- The thumbprint of the certificate wasn't added to the wif.config file on the AOS machine. To fix the issue, see step 8 in the Run a single-user test by using the Performance SDK section for information about how to add the certificate to the wif.config file. After you modify the wif.config file, be sure to restart the application through Service Fabric Explorer.

**MS.Dynamics.Test.Team.Foundation.WebClient.InteractionService.dll.config is missing from the deployment items**

This issue usually occurs only when you run load tests.

**Error example**

```xml
<Test class name>.TestSetup threw exception. System.InvalidOperationException: Could not find endpoint element with name 'ClientCommunicationManager' and contract 'Microsoft.Dynamics.Client.InteractionService.Communication.Reliable.IReliableCommunicationManager' in the ServiceModel client configuration section. This might be because no configuration file was found for your application, or because no endpoint element matching this name could be found in the client element..
at System.ServiceModel.Description.ConfigLoader.LoadChannelBehaviors(ServiceEndpoint serviceEndpoint, String configurationName)
```

**Solution**

This issue occurs when the system can't find the MS.Dynamics.Test.Team.Foundation.WebClient.InteractionService.dll.config file when the load tests are run, because the file wasn't added as a deployment item. Verify that the MS.Dynamics.Test.Team.Foundation.WebClient.InteractionService.dll.config file is in the Out folder for the test run:

```xml
<solution path>\TestResults\<your test run>\Out
```

If the file is missing, add it to the deployment items in the test settings.

**IMPORTANT**

There are two files that have very similar names. The name of one file ends in *.dll*, and the name of the other file ends in *.dll.config*. The *.dll.config* file must be in the deployment items in the test settings.

**CloudEnvironment.Config is missing from the deployment items**

This issue usually occurs only when you run load tests.

**Error example**

```xml
```

**Solution**

This issue occurs when the CloudEnvironment.Config file isn't present when the tests are run. The issue typically occurs when you run load tests and the CloudEnvironment.Config file wasn't added as a deployment item. Verify that the CloudEnvironment.Config file is in the Out folder for the test run:

```xml
<solution path>\TestResults\<your test run>\Out
```

If the file is missing, add it to the deployment items in the test settings.
InteractiveClientId wasn't specified in the settings

Error example

```
Microsoft.CE.VaultSDK.SecretProviderException: InteractiveClientId was not specified in settings
```

Solution

This issue occurs when no value is specified for the `SelfSigningCertificateThumbprint` key in the CloudEnvironment.Config file. In the CloudEnvironment.Config file, find the following line, and paste in the thumbprint of the certificate that you created and installed.

```
<ExecutionConfigurations Key="SelfSigningCertificateThumbprint" Value="" />
```

The remote host forcibly closed an existing connection

Error example

```
System.ServiceModel.CommunicationException: An error occurred while making the HTTP request to <Host name>/Services/AxUserManagement/Service.svc/ws2007FedHttp. This could be due to the fact that the server certificate is not configured properly with HTTP.SYS in the HTTPS case. This could also be caused by a mismatch of the security binding between the client and the server.** ---> System.Net.WebException: The underlying connection was closed: An unexpected error occurred on a send. ---> System.IO.IOException: Unable to read data from the transport connection: An existing connection was forcibly closed by the remote host. --->
System.Net.Sockets.SocketException: An existing connection was forcibly closed by the remote host.
```
Solution
Run the following Windows PowerShell script on the development machine.

Set-ItemProperty HKLM:\SOFTWARE\Microsoft\.NETFramework\v4.0.30319 -Name SchUseStrongCrypto -Value 1 -Type dword -Force -Confirm:$false
if (((Test-Path HKLM:\SOFTWARE\Wow6432Node\Microsoft\NETFramework\v4.0.30319)))
{
    Set-ItemProperty HKLM:\SOFTWARE\Wow6432Node\Microsoft\NETFramework\v4.0.30319 -Name SchUseStrongCrypto -Value 1 -Type dword -Force -Confirm:$false
}

The w3svc service wasn't found on the computer
This error only occurs when you run load tests by using Microsoft Visual Studio Online.

Error example

--->
System.InvalidOperationException: Service w3svc was not found on computer '.'.
--->
System.ComponentModel.Win32Exception: The specified service does not exist as an installed service

Solution
A hotfix is available that resolves this issue. The Microsoft Knowledge Base (KB) number is 4095640.
This topic explains how you can use the Trace parser to consume traces and analyze performance in your deployment. You can use the Trace Parser to find and diagnose various types of errors. You can also use the tool to visualize execution of X++ methods, as well as the execution call tree.

**NOTE**

There are many more features in the Trace parser are similar to Microsoft Dynamics AX 2012. See the [Dynamics Ax Performance Team Blog](#) for more information.

### Finding the Trace parser

Trace parser should be preinstalled with your developer deployment or VHD. The install location is here: `C:\Program Files (x86)\Microsoft Dynamics Trace Parser`. If it's not installed, you can run the installer from `C:\PerfSDK\PerfTools\traceparser.msi`.

### Capturing events

There are two ways that you can obtain the data that you will analyze in the Trace parser. They include:

- Capture events from the local installation.
  - If the **Select Trace** window isn’t already open, go to the **File** menu and click **Open trace**. In the **Select Trace** window, click **Capture Events**. After selecting your providers, click **Start**. The Trace Parser tool will start listening to all the providers and capturing the events. Capturing stops when you click **Stop and Import**.
  - Open an existing ETL (Windows Event) file that was captured using tools such as Logman.

### Viewing traces

**Timeline view** The Timeline tab is the first tab that you see after you import a trace into the Trace Parser. This tab is shown in the following illustration.
The **Timeline** tab has the following major components:

- The **Select Grouping** drop-down allows you to group based on a variety of categories, such as Customer ID, Username, Session Name, etc. Groupings will display maximum and minimum timestamp of events, total number of events, and lowest event level within the grouping.
- List of all events in a threaded or unthreaded view.
- Property grid displayed for the selected event.
- Timeline chart for all the selected events.
- Filtering of events.
- Session analysis notes.

**Call tree view** By selecting the **Call Tree** tab, you can see the call tree for all X++ methods. The tab is shown below.

![Example of information shown in the Call Tree tab](./media/3_desktop.png)

Similarly, you can display the **X++** tab to view a list of all the X++ methods. They will be sorted by fields such as Inclusive/Exclusive durations, RPC, or Database calls. Note that these are similar to the corresponding tabs in Trace Parser and have the same behavior.

**Additional resources**

*Develop and customize home page*
This topic provides an overview of the Performance timer, which is a tool that helps you to determine why your system’s performance might be slow.

To open the Performance timer, open your webpage with the added parameter debug=develop:
https://yoursite.cloud.test.dynamics.com/en/?cmp=USMF&debug=develop Note: When you run in debug mode you will notice slower performance. You can quickly get an overview of most performance issues by pressing F12 and working with the debugging tools that are available in your browser. The timer will show up here.

To open a list page, for example, such as the purchase order list page, click the Performance timer. The following screenshot shows the separation between client time and server time, and the total time. Additionally, you can see a set of performance counters and expensive server calls.

For more information about the server performance counters, click on any of the links.

- **Forms** - Forms will show how many forms are currently open, plus the rate at which they opened and closed (per second), and a set of counters, such as the total amount of created or closed forms.
- **GC** - This is information about the garbage collection processes on the server.
- **Web client session** - This shows how many web client sessions you currently have and how many are in use.
- **Services Session provider** - This is the total number of sessions created.

For more information, click a link. In the next screen, you can see how many SQL queries were triggered by this individual call and which SQL query was the most expensive.
This information can help you to understand what to trace and where to start troubleshooting.
This tutorial shows you how to create and run test cases.

**Prerequisites**

You will need to deploy Developer Topology with Developer and Build VM.

**Key concepts**

- Use SysTest Framework to author unit/component test code.
- Test isolation
- Test module creation to manage test code and FormAdaptors.
- Import Task Recorder recordings into Visual Studio to generate test code.
- Integrate a Test module with a build machine.

**Use SysTest Framework to author unit/component test code**

You can create new test cases to test the functionality in an application.

1. Open Visual Studio as an administrator.

2. On the File menu, click Open > Project/Solution, and then select FleetManagement solution from the desktop folder. If the solution file is not on your computer, the steps to create it are listed in Tutorial: Create a Fleet Management solution file out of the Fleet Management models in the AOT.

3. In Solution Explorer, right-click the Fleet Management solution, point to Add, and then click New Project.

4. Choose Finance and Operations as the project type to create.

5. Name this new project FleetManagementUnitTestSample, specify the FleetManagement folder on the desktop (C:\Users\Public\Desktop\FleetManagement) as the location, and then click OK.

6. In Solution Explorer, right-click the new project, and then click Properties.

7. Set the Model property to FleetManagementUnitTests, and then click OK.
8. Right-click the FleetManagementUnitTestSample project, point to Add, and then click New Item.

9. In the Add New Item window, select Class as the type of element to add. Name the new class FMUnitTestSample, and then click Add.

10. In the first line of the code for the new class, indicate that the class extends the SysTestCase class.

11. Add the following code to define the methods for the class. These methods define two additional tests.
class FMUnitTestSample extends SysTestCase
{
  public void setup()
  {
    // Reset the test data to be sure things are clean
    FMDataHelper::main(null);
  }

  [SysTestMethodAttribute]
  public void testFMTotalsEngine()
  {
    FMRental rental;
    FMTotalsEngine fmTotals;
    FMRentalTotal fmRentalTotal;
    FMRentalCharge rentalCharge;
    FMRentalTotal expectedTotal;
    str rentalID = '000022';

    // Find a known rental
    rental = FMRental::find(rentalID);

    // Get the rental charges associated with the rental
    // Data is seeded randomly, so this will change for each run
    select sum(ExtendedAmount) from rentalCharge
      where rentalCharge.RentalId == rental.RentalId;

    fmTotals = FMTotalsEngine::construct();
    fmTotals.calculateRentalVehicleRate(rental);

    // Get the totals from the engine
    fmRentalTotal = fmTotals.totals(rental);

    // Set the expected amount
    expectedTotal = rental.VehicleRateTotal + rentalCharge.ExtendedAmount;

    this.assertEquals(expectedTotal, fmRentalTotal);
  }

  [SysTestMethodAttribute]
  public void testFMCarValidateField()
  {
    FMCarClass fmCar;

    fmCar.NumberOfDoors = -1;
    this.assertFalse(fmCar.validateField(Fieldnum("FMCarClass", "NumberOfDoors")));

    fmCar.NumberOfDoors = 4;
    this.assertTrue(fmCar.validateField(Fieldnum("FMCarClass", "NumberOfDoors")));
  }
}
Test isolation

For a test to be of high value it must be reliable. A test will pass or fail consistently, independent of other factors such as other tests. One typical cause of unreliable tests is leaking state, such as data left behind in the database that influences downstream tests. To prevent this type of issue, you can use the `SysTestTransaction` attribute.

<table>
<thead>
<tr>
<th>TESTTRANSACTIONMODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoRollback</td>
<td><strong>Default.</strong> This provides the best isolation. All transactions are rolled back using SQL save points, and all database statements are routed to the main connection, including user connections. No data will be persisted.</td>
</tr>
<tr>
<td>LegacyRollback</td>
<td>All insert statements are tracked and deleted during cleanup. All insert statements are downgraded to row-by-row. One typical use case is when testing user connections or concurrency scenarios. This isolation level will clean up setup data, and the recommendation is to wrap each test method in a ttsBegin and ttsAbort.</td>
</tr>
<tr>
<td>LegacyRollbackWithUpdateTracking</td>
<td>All update, delete, and insert statements are tracked and reverted during cleanup. All insert, update, and delete statements are tracked and downgraded to row-by-row. This is the slowest isolation level.</td>
</tr>
<tr>
<td>None</td>
<td><strong>Only use for debugging.</strong> This provides no isolation. This setting can be useful to temporarily debug a test, as it allows you to use the regular user interface to navigate the data that the test created.</td>
</tr>
</tbody>
</table>

Example:

```java
[SysTestTransaction(TestTransactionMode::LegacyRollback)]
class MyTestSample extends SysTestCase
```

Test module creation to manage test code and FormAdaptors
Creating a test specific module helps to keep test code together and manageable.

1. Open Visual Studio and go to Dynamics 365 > Model Management > Create model.

2. Enter the model name, select the layer, and then enter any additional details. Note that it’s a good idea to include the word Test in the name of the test module. The default build definition is configured to discover all test modules that contain the word Test.

3. Because this model holds forms from the Application Platform/Foundation, add references to models shown below.

![Create model dialog](image)

After the base test module is in place, you can import a Task Recorder recording to generate test code. When you import a Task Recorder recording XML, test code is generated using FormAdaptors. Form adaptors are wrapper classes over forms which provide strongly typed API that can be used to test form functionality. We have included pre-generated FormAdapters for each package for built-in forms. In the test module, add a reference to the corresponding Form Adaptor for packages and Test Essentials, which has helper methods to execute test code.

**Import a Task Recorder recording into Visual Studio to generate test code**

You can generate test code from Task Recorder recording to execute headless (non-UI) test.

1. Record a scenario in by using Task Recorder.

2. To import a Task Recording, in Visual Studio, click Dynamics 365 > Addins > Import Task Recording.

3. In the Import Task Recording dialog, select the Test Module (ISVTestModule) under which you want to import task recording, and browse to recording xml file.
4. The task recording import process generates test code that is based on the SysTestAdapter and FormAdaptor which can be viewed in Visual Studio IDE. We do not expect you to change any test source code that is generated as part of this step.

5. After the test code is generated, set up Visual Studio options for test discovery and execution:
   - If you have a 64-bit machine, you can run unit tests and capture code coverage information as a 64-bit process.
   - To configure this, select Test > Test Settings > Default Processor Architecture, and then select X64.
   - You might run into a situation in which the test execution engine opens and locks an assembly in your test project. When this happens, you can’t for example, save changes to the assembly. To fix this, select Test > Test Settings, and then select Keep Test Execution Engine Running.
   - Now that you have test code generated in Visual Studio IDE, it’s time to discover the test and try executing them locally.

6. From menu options, select Test > Windows, and then click Test Explorer. After the Test Explorer window is open, it will try to discover test from test code and list all the available tests as shown below.

7. Select the test and then click Run > Execute selected. This will execute test against the locally deployed environment.
Integration of the test module with build process

After the test module is a part of source control, the build process template will discover all test modules, which contain the word Test in the name. The following illustration shows build and test execution as part of Visual Studio Online.
CTP5.000.00_03.18.15_18.11.18_20150318.1 - Build succeeded

View Summary | View Log - Open Drop Folder | Diagnostics | <No Quality Assigned> | Actions

Latest Activity
Build test modified by user 4 seconds ago.

Request Summary
Request 1, requested by user 23.7 minutes ago, completed.

Summary

Default Configuration and Platform
0 errors(), 0 warning(s)

Test Run
SYSTEM@RXP007C62RAINS 2015-03-18 18:41:11, 1 of 1 test(s) passed
SYSTEM@RXP007C62RAINS 2015-03-18 18:41:34, 3 of 4 test(s) passed

No Code Coverage Results

Impacted Tests
No tests were impacted.

<table>
<thead>
<tr>
<th>Test Name</th>
<th>ID</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>OneTest2.testMethod</td>
<td>Dynamics.AX.App</td>
<td></td>
</tr>
<tr>
<td>OneTest2.testMethod</td>
<td>Dynamics.AX.App</td>
<td></td>
</tr>
<tr>
<td>OneOverlayTest.testMethod</td>
<td>Dynamics.AX.App</td>
<td></td>
</tr>
<tr>
<td>OneOverlayTest.testFailMethod</td>
<td>Dynamics.AX.App</td>
<td></td>
</tr>
</tbody>
</table>
This topic describes the options for testing in Visual Studio.

A custom unit test adapter is available in Visual Studio. This adapter lets test authors use the standard Test Explorer window in Visual Studio to schedule X++ tests and analyze test results. Developers can author tests by using SysTestAdaptor. They can also generate test code from Task Recorder recordings. These test cases can then be added to build systems for validations.

Author unit/component test code by using the SysTest Framework

When you create a project in Visual Studio, you can add an X++ unit test. You extend the class with SysTestCase, and then either add the SysTestMethodAttribute attribute or prefix the case with "test" in the method name.

```csharp
class FMUnitTestSample extends SysTestCase
{
    [SysTestMethod]
    public void testTotalsEngineConfig()
    {
    }
}
```

After you save the class, each test appears in Test Explorer, just as a C# test would appear.

In Test Explorer, you can run the tests, or you can debug the test case by right-clicking and running or debugging the selected tests.
Before you can run tests, you must build the project so that it includes tests.

You can also discover existing tests for an object in your project. Discovery uses cross-reference data. Right-click an object in the project, and then select Discover Related Tests. This command queries the cross-reference data and returns any tests that reference the object. The list of test cases is displayed in Test Explorer.

By using this functionality, you can run all the relevant tests. Test Explorer contains all tests for the current project and all tests for the referenced objects.

**Generate test code by importing Task Recorder recordings into Visual Studio**

You can import the XML for Task Recorder recordings to generate test code that can be used to validate various business process scenarios.
Generated code is based on the SysTest Framework and FormAdaptors. FormAdaptors are wrapper classes over pages. They provide strongly typed application programming interfaces (APIs) that can be used to test page functionality. Pre-generated FormAdaptors are included for each package for built-in pages. In a test module, add a reference to a corresponding FormAdaptor for packages and “Test Essentials,” which contain helper methods to run test code.

**Advanced Options**

For advanced options to categorize and filter tests for execution, see [SysTest Filtering using class and method attributes](#).
This topic describes how to deploy and use an environment that supports continuous build and test automation.

**Prerequisites**

Cloud deployment of virtual machines (VMs) requires a Microsoft Azure DevOps subscription.

**Workflow**

After you configure an Azure DevOps subscription in Microsoft Dynamics Lifecycle Services (LCS), you can use LCS to deploy developer VMs or build/test VMs. LCS configures a developer VM that can be mapped to an Azure DevOps project. LCS also configures a build VM that is automatically mapped to an Azure DevOps project and has a build agent/controller that builds modules from the Azure DevOps project and runs automated tests that have an external endpoint for validation. The following illustration shows a typical workflow.

![Workflow for Cloud-hosted development and build/test environments](image)

This workflow includes an LCS deployment of a developer VM and a build/test VM in Azure.

- LCS creates developer and the build/test environments in Azure. To create a build/test environment, LCS must be able to determine where the source code for the Azure DevOps project is.
- The developer works on source code on the developer VM, and the work is synced to the Azure DevOps project.
- The build process synchronizes the code from Azure DevOps onto the build/test VM and produces deployable packages that you can apply to sandbox and production environments. The source code doesn't flow directly from the development VM to the build/test VM. They are synced through Azure DevOps.

For information about how to write custom test code or generate automated test code to integrate with the build infrastructure, see [Testing and validations](#).

**Set up Azure DevOps**

**Choose a plan**

The first step is to [choose an Azure DevOps plan](#) for your organization.
Set up Azure DevOps

To set up Azure DevOps, follow these steps.

1. Create a personal access token. The token is used for all LCS background actions. These actions include upgrade and deployment. When users initiate actions from LCS, LCS expects that those users will be added to Azure DevOps. The users must authorize LCS access to Azure DevOps on their behalf.

2. Configure LCS.

Until you authorize LCS access to Azure DevOps, you will see a “setup is not complete” message in action center.

Suspend current builds

If you're deploying a build environment on an existing Azure DevOps project that already has a build definition, make sure that you don't have any active triggers to queue the build. Additionally, make sure that no builds are scheduled or queued against the build pool.

Deploy Developer and Build/Test environments from LCS

LCS provides an option to deploy Development and Build/Test environments. With this option, you can deploy developer and build VMs in the cloud that are connected to your Azure DevOps project.

Azure DevOps credential setup and linking to LCS project

If you have not already done so, you need to first setup your LCS project to connect to your Azure DevOps project before you deploy a build environment.

1. Login to the LCS portal to connect to Azure DevOps and your LCS project at https://lcs.dynamics.com/.
2. Select a project that you are working on.
3. Click the Project Settings tile.
4. Select Azure DevOps and enter the Azure DevOps URL where the source code for your module project is located.
5. Specify the Azure DevOps link, authorize, and then click Choose default project.

Check-in migrated or new module code into Azure DevOps

As part of code Migration process or development activities, we expect you to check-in your model source files and the associated test model source files into Azure DevOps. If you have migrated your code using the LCS migration service, this is automatically done for you. If you have not checked in any code into Azure DevOps and work on direct check-in, you must follow certain guidelines for the Azure DevOps folder structure. This will help with setting up correct build definition. All modules should be added to root folder Metadata. Under each module, there should be two folders. One folder contains all models. The other folder should contain descriptor XML for that module.
Deploy a Build environment

The topic [Deploy and access development environments](#) describes how to deploy developer environments. Use the same flow to deploy a build environment. As you are going through the deployment or configuration wizard, when prompted to **Select a Topology**, select **DevTest** then select a **Build and Test** topology.

As part of the deployment wizard, you can configure the build agent name and build agent pool.

Click **Advanced settings**, select **Azure DevOps**

1. **Build Agent Name**: Friendly name for build agent on Azure DevOps
2. **Build Agent Pool**: specify build agent pool name which should be used for build machine deployment. Make sure Azure DevOps contains at least one agent pool. By default, there will be the default pool. If you have deleted the default pool then build deployment will fail.
3. **Branch Name**: Specify your Azure DevOps source code branch which will be default source code sync location for the build VM. Default branch is “Main”.

Test integration with the build

There are two ways to integrate test as part of build process for testing and validation:

- SysTest framework based unit and component level tests.
- Generate code from Task Recorder recording XML for automated test execution.

The details of these two approaches are mentioned in the [Testing and validation](#) article. Review this article for testing and validation strategy.

Use the Build VM environment

When a Build VM is deployed in Developer topology through LCS, it is pre-configured and ready to start a build. You can change the default configuration at any time from the Visual Studio IDE or the Azure DevOps interface. On a Build VM, the module source code is synchronized to the build machine for easy build setup. The build machine is also auto-configured with default settings for build agent, build controller, build process template, and build definition. Tests that are integrated with build definition are executed after the build is successful.
Review a pre-configured customizable build environment

The build VM contains the vNext build agent which was released as part of Azure DevOps. When you deploy the Build VM, the build agent is configured by default to connect and sync with the Azure DevOps project. As a part of the Build VM configuration, the default build definition is also created and configured, as shown below.

Default build definition contains multiple tasks to perform specific operation, as described below.

1. Configure the predefined variables parameters that will be passed to the build. To set up a clean database for every build execution, provide the name of the database backup file for the `DatabaseBackupToRestore` variable. The packages folder is restored at every build with a copy of a clean package folder.

2. Build the solution to discover and build all modules under "Trunk/Main" branch as shown below.
3. Use “Deploy Report” task to generate reports and deploy on build VM.

4. Use “Database Sync” task to synchronize the database to local SQL on build VM.

5. After the build is successful, create a deployable package that can be used to update sandbox/staging environment.

6. “Copy and publish build artifacts” uploads the deployable package to Azure DevOps artifacts location.

7. For test execution, there are three default tasks “Test Setup”, “Execute Test” and “Test End”.
8. The default build is scheduled to trigger start every day at 5 PM. You can change trigger as per your team’s need to "Continuous" for each check-in.

You can make changes to the default configuration, and the build VM will be ready to trigger a build.

Start a build and verify the build and test execution results

After you review the default build configuration, you can manually trigger a build from Visual Studio IDE or Azure DevOps web interface.

1. Open your browser and connect to the Azure DevOps URL.
2. Login using your credentials.
3. On the home page, under Recent projects and solutions, select a project.
4. From top links options, select BUILD.
5. On the left panel, select the default build definition instance.
6. Right-click and select Queue Build to trigger a build for your module and test module that is already checked into the Azure DevOps source control.

Success or failure for the build will display, as shown by the following examples. View all builds.
Select specific completed build and view success/failure details.

Click on Test link to visualize test execution failure.
Starting with Platform update 12, the SysTest framework contains improvements to the SysTest class and method attributes in X++.

These improvements also change how these attributes work in the Visual Studio test window as well as the Visual Studio Test Console, which is the tool used in the automated build process. The SysTest framework now supports the major test attributes in the adaptor to be on par with the MSTest framework adaptor. This includes attributes like **Category**, **Owner**, **Priority**, and **Test Property**.

### TestCategory

The **Category** attribute, **SysTestCategory**,** was already available in previous platform updates using the **SysTestCategory** attribute. Starting with Platform update 12, you can specify multiple categories on both the class level and the individual method level. Additionally, **TestCategory** is enabled for filtering in the Visual Studio Test Console. This means that you can create build pipelines with test filters on specific categories. You can use the **TestFilter** variable in the build pipeline. For example, to run tests only with a category **Nightly**, set the variable to **TestCategory=Nightly**.

### Priority

The **Priority** attribute **SysTestPriority**, which requires an integer value, is now available. A priority can only be specified once, but is supported on both the class and method level, with method level taking precedence over class level. The priority is also exposed as a test filter. This means that you can use the **TestFilter** variable in the build pipeline. For example, to only run tests with a priority of **1**, set the variable to **Priority=1**.

### Owner

The **Owner** attribute, **SysTestOwner**, has also been added. This attribute was technically already supported for filtering in the **Test Toolbox** window, but the attribute itself was missing in X++. Similar to **Priority**, an owner can only be specified once and is supported on both the class and method level, with the method level taking precedence. **Owner** is not available as a test filter for the console, which aligns with the MSTest adaptor for Visual Studio Test Console. However, **Owner** will appear in the **Test Toolbox** window in Visual Studio.

### Test Property

The **Test Property** attribute, **SysTestProperty**, has existed in previous releases of the platform, but wasn't fully functional. Unfortunately, the **SysTestProperty** attribute exists in the **Application Foundation** package as opposed to the other attributes which exist in the **Test Essentials** package. Because of our commitment to backward compatibility of the platform, we currently cannot move this attribute to its expected location, so your code will need a reference to the **Application Foundation** package to use it. **SysTestProperty** specifies a property and a value (two strings), and can now be used in the **Test Toolbox** window in Visual Studio. **Test Property** can be specified multiple times, and can exist on both the class and method level. **Test Property** is not available for test filtering in the Visual Studio Test Console, in line with the MSTest adaptor.

For advanced filtering syntax that can be used with the Visual Studio Test Console and to review the filtering example for the MSTest framework, see **Running selective unit tests**.
NOTE
For test filtering purposes, the SysTest framework allows you to filter on `FullyQualifiedName` and `Name`. 
The Acceptance test library (ATL) is an X++ test library that offers the following benefits:

- It lets you create consistent test data.
- It increases the readability of test code.
- It provides improved discoverability of the methods that are used to create test data.
- It hides the complexity of setting up prerequisites.
- It supports high performance of test cases.

### Example of a test that is written in ATL

```plaintext
// Create the data root node
var data = AtlDataRootNode::construct();

// Get a reference to a well-known warehouse
var warehouse = data.invent().warehouses().default();

// Create a new item with the "default" setup using the item creator class. Adjust the default warehouse
// before saving the item.
var item = items.defaultBuilder().setDefaultWarehouse(warehouse).create();

// Add on-hand (information about availability of the item in the warehouse) by using the on-hand adjustment
// command.
onHand.adjust().forItem(item).forInventDims([warehouse]).setQty(100).execute();

// Create a sales order with one line using the sales order entity
var salesOrder = data.sales().salesOrders().createDefault();
var salesLine = salesOrder.addLine().setItem(item).setQuantity(10).save();

// Reserve 3 units of the item using the reserve() command that is exposed directly on the sales line entity
salesLine.reserve().setQty(3).execute();

// Verify inventory transactions that are associated with the sales line using the inventoryTransactions
// query and specifications
salesLine.inventoryTransactions().assertExpectedLines(
    invent.trans().spec().withStatusIssue(StatusIssue::OnOrder).withInventDims([warehouse]).withQty(-7),
    invent.trans().spec().withStatusIssue(StatusIssue::ReservPhysical).withInventDims([warehouse]).withQty(-3));
```

### Concepts

The structure and naming of the classes and methods in ATL are quite rigid. This rigidity helps improve
discoverability and also makes it easier to write tests, even in domains that you're unfamiliar with.

The classes are grouped into the following concepts:

- **Navigation** – Discover entities and test data methods in a familiar hierarchy.
- **Test data methods** – These methods are used to set up test data.
- **Entities** – Entities represent data and associated behavior that is perceived as a single unit.
- **Creators** – Creators let you create specific test data.
- **Commands** – Commands run business operations.
- **Queries** – Queries find entities.
Specifications – Specifications describe expected entities at the end of the test.

Code generation

Queries and specifications help simplify the process of creating entities. For more information, see Acceptance test library Code generation wizard. The Code generation wizard can be used to create scenarios and update them.

Further reading

Microsoft has used ATL internally for several years, as the foundation for thousands of tests. For more information see, Best practices for the Acceptance test library and Acceptance test library FAQ.
To simplify the discoverability of generation methods for test data, a set of navigation objects is introduced. For more information about the generation methods, see Test data methods.

Navigation should start from the root object, the module must be specified, and then the entity must be specified together with the test data methods.

```java
data.module().entity().testDataMethod();
```

### Examples

```java
modelGroup = data.invent().modelGroups().fifo();

itemBuilder = data.products().items().whsBuilder();

data.sales().salesOrders().ensureCanCreate();

data.invent().parameters().enableQualityManagement();
```

### Advantages

- Discoverability of data creation application programming interfaces (APIs). IntelliSense helps you get to the helper methods that are relevant for the entity.
- Local references to frequently used nodes (for example, `items.whsBuilder()`). Therefore, long names aren’t required.

### Root navigation object

The root navigation object is the starting point for finding the test data methods that are required. The root navigation object exposes modules where test data methods are defined.

#### Variable naming

The suggested name for variables that reference the navigation root is `data`.

#### Class naming

The root class is named `AtlDataRootNode`.

### Module navigation object

The module navigation objects let you group test data methods by relevant modules. Module navigation objects can expose entity navigation objects.

#### Navigation node naming

Module names should be based on the names of the modules on the main menu. However, a short version or an abbreviation should be used to support brevity of test code.

### Examples

- **Sales and marketing** module: `data.sales()`
- **Inventory management** module: `data.invent()`
Class naming

AtlData<ModuleName>

Examples

- **Sales and marketing** module: `AtlDataSales`
- **Procurement and sourcing** module: `AtlDataPurch`
- **Inventory management** module: `AtlDataInvent`
- **Accounts receivable** module: `AtlDataCust`
- **Accounts payable** module: `AtlDataVend`

Entity navigation objects

Entity navigation objects let you group test data methods by relevant entities.

Navigation node naming

The plural of the entity name should be used as the name of the entity navigation node.

Examples

```java
data.products().items();
data.whs().warehouses();
```

Class naming

AtlData<ModuleName>EntityNamePlural

Examples

```java
AtlDataProductsItems
AtlDataInventChargeGroups
```

Notes

The same entity navigation object can be exposed from multiple modules when this approach makes sense. For example, `AtlDataProductsItems` is exposed from both `data.product()` and `data.invent()`.

Helper navigation objects

Sometimes, a test data method isn't specific to any entity. In this case, a helpers node can be exposed at the module level.

Navigation node naming

The helpers navigation node should be named `helpers`.

Examples

```java
data.helpers();
data.whs().helpers();
```
AtlDataHelpers

AtlDataWSPHelpers
Entity and helper navigation objects expose test methods that let you set up test data. This topic provides information about the most common types of test data methods.

Factory methods

Factory methods focus on creating data that doesn't yet exist in the database. There are two types of entity factory methods, `init` methods and `create` methods. An `init` method initializes the entity but doesn't save it to the database. A `create` method initializes the entity and saves it to the database.

**Naming convention**

```
init<EntitySpecification>
create<EntitySpecification>
```

In this naming convention, `<EntitySpecification>` is the description of the key characteristics of the object that must be created.

**Examples**

```
salesOrder = data.sales().salesOrders().initDefault();
purchaseOrder = data.purch().purchaseOrders().createDefault();
```

**Best practices**

The `create` method should always call the `init` method that has the same entity specification.

**Example**

```
public AtlEntitySalesOrder createDefault()
{
    AtlEntitySalesOrder salesOrder = this.initDefault();
    salesOrder.save();
    return salesOrder;
}
```

**Prerequisite data**

The `init` method should take care of setting up prerequisites.

Before some entities can be created, specific prerequisites must be set up. In these cases, the `ensure` method must be called before the entity is initialized. You must also subscribe to all the entity events that require automatic setup of prerequisites.

**Example**
public AtlEntitySalesOrder initDefault()
{
    AtlEntitySalesOrder salesOrder;

    this.ensureCanCreate();

    salesOrder = new AtlEntitySalesOrder();
    salesOrder.parmCustomer(data.cust().customers().default());

    _salesOrder.postingInvoice += eventhandler(this.ensureCanPostInvoice);
    _salesOrder.postingPackingSlip += eventhandler(this.ensureCanPostPackingSlip);
    _salesOrder.releasingToWarehouse += eventhandler(this.ensureCanReleaseToWarehouse);
}

Builder methods

Builder methods are responsible for initializing creator objects that will be used to create data that doesn't yet exist in the database.

**Naming convention**

<EntitySpecification>Builder

In this naming convention, `<EntitySpecification>` is the description of the key characteristics of the object that must be created.

**Example**

```java
catchWeightItem = data.invent().items().cwBuilder();
```

Well-known data methods

Well-known data methods provide a way to reference an entity that is set up in a specific way. If the entity doesn't exist in the database, it's created.

**Naming convention**

<EntitySpecification>

In this naming convention, `<EntitySpecification>` is the description of the key characteristics of the object that must be retrieved.

**Example**

```java
fifo = data.invent().modelGroup().fifo();
```

In this example, the contract of the method specifies that the model group should use first in, first out (FIFO) as the inventory model. The rest of the settings can be left at their default values.

Sometimes, a real-world name communicates the contract better.

```java
pieces = data.common().units().pieces();
```

In this example, it's clear that `pieces` is a unit of measure of the "quantity" class, and that is has a decimal precision of 0 (zero).

**Contract of well-known data methods**
Here are a few things to remember about the common contract of the well-known data methods.

- Two calls to the same well-known data method should provide the caller with the reference to the same entity.

```java
fifo1 = data.invent().modelGroups().fifo();
fifo2 = data.invent().modelGroups().fifo();
fifo1.InventModelGroupId == fifo2.InventModelGroupId;
```

- Creation of a test entity isn’t always worth the effort. If a test entity isn’t created, the corresponding record buffer should be returned from the well-known data method. For example, if you don’t invest the time and effort to create the Site entity, the `site` well-known data method will return InventSite records.

```java
InventSite site = data.invent().sites().default();
```

- Well-known data methods can take IDs as optional parameters when this approach makes sense.

```java
item1 = data.products().items().default('Item1');
item2 = data.products().items().default('item2');
```

### Implementation

If there is already a builder or a factory method that is named `<EntitySpecification>`, it should be used as the internal implementation to create the well-known entity.

#### Example

```java
public InventTable whsBatchAbove(ItemId _itemId = this.whsBatchAboveItemId())
{
    InventTable whsItem = InventTable::find(_itemId, true);
    if (!whsItem)
    {
        whsItem = this.whsBatchAboveBuilder().setItemId(_itemId).create();
    }
    return whsItem;
}
```

### Ensure methods

Ensure methods are responsible for setting up prerequisites that are required in order to create an entity or run a business operation.

#### Naming convention

`ensureCan<ExecuteBusinessOperation>`

In this naming convention, `<ExecuteBusinessOperation>` is a verb that describes the business operation.

#### Examples

```java
data.sales().salesOrders().ensureCanCreate();
data.purch().purchaseOrders().ensureCanPostProductReceipt();
```

### Implementation

Figuring out prerequisites for a complex business operation, such as invoice posting, can be complicated and requires lots of knowledge about the feature area.
Example

```java
public void ensureCanCreate()
{
    data.helpers().setNumberSequenceReference(extendedTypeNum(InventDimId));
}
```

**Automatic prerequisite setup**

To enable automatic prerequisite setup, you must call the `ensure` methods in the appropriate `init` method. For more information, see the Factory methods section earlier in this topic.

**Query methods**

Query methods are responsible for initializing new queries for the entity type of the navigation node that they are defined on.

**Naming convention**

`query`

**Examples**

```java
loadLinesQuery = data.whs().loadLines().query();
purchaseLinesQuery = data.invent().transferOrderLines().query();
```

**Specification methods**

Specification methods are responsible for initializing new specification objects for the entity type of the navigation node that they are defined on.

**Naming convention**

`spec`

**Example**

```java
loadLinesSpec = data.whs().loadLines().spec();
```

**Find methods**

Find methods let you find an entity based on the primary key.

**Naming convention**

`find`

**Example**

```java
salesOrder = data.sales().salesOrders().find(salesOrderId);
```

**Automatic prerequisite setup**

If the entity has query support, the implementation should use the query that has already set up prerequisite support. Otherwise, after you find the record buffer and initialize the new instance of the entity, you should subscribe `ensure` methods to the business operation events of the entity.
A test entity class represents data and behavior that are perceived as a single concept. Test entity classes are based on pages such as Sales order, Transfer order, and Released product. The test entity classes expose the properties that are most often used in test scenarios, and the behavior that is most important from the perspective of test data setup and scenario tests.

An entity in the Acceptance test library (ATL) must have the following methods:

- Property methods that are used to get and set entity properties.
- Fluent setter methods that enable entity properties to be set in a fluent manner.
- A method that saves the entity to the database.

An entity in ATL can have the following methods:

- Action methods that are used to expose business operations that are relevant to the entity.
- Query methods that are used to enable navigation to components and related entities.

Naming convention

\[
\text{[AtlEntity]}<\text{ModuleName}><\text{EntityName}>
\]

In this naming convention:

- \(<\text{ModuleName}>\) is based on the names of the modules in main menu. You should use a short version or an abbreviation to support brevity of test code.
- \(<\text{EntityName}>\) is based on the user interface (UI) names instead of the table names. For example, use SalesOrder, not SalesTable.

If an entity has two UI names, it's OK to use the shorter name. For example, you can use Item instead of ReleasedProduct, because these names are used interchangeably.

Examples

- AtlEntitySalesOrder
- AtlEntityTransferOrderLine

Property methods

One of the main purposes of a test entity is to expose data. The properties of the entity can be set or retrieved by using \(\text{parm}\) (property) methods.

**Primitive type properties**

Create a \(\text{parm}\) method to expose a primitive type property.

Example
public SalesQty parmQuantity(SalesQty _qty = 0) {
    if (!prmisDefault(_qty)) // setter
    { 
        salesLine.SalesQty = _qty;
        this.onSalesQtyOrInventDimChange();
    }
    return salesLine.SalesQty;
}

Entity references
If there is a customer entity that is named AtlEntityCustomer, for example, a reference to customer should be exposed as a property method on the AtlEntitySalesOrder entity.

public AtlEntityCustomer parmCustomer(AtlEntityCustomer _custTable = null)

The property method can be used as either a setter or a getter.

salesOrder.parmCustomer(customer); // setter

customer = salesOrder.parmCustomer(); // getter

Entity reference methods naming conventions
The parm prefix should be used to identify property methods. When you expose an entity reference property, use the UI name of the field instead of the Application Object Tree (AOT) name. If the UI name includes the Id, Code, or Number suffix, omit the suffix. For example, use parmItem instead of parmItemNumber.

Record references
If the customer entity hasn't yet been created and won't be created in the near future, the reference property should expose the corresponding record buffer (CustTable).

public CustTable parmCustomer(CustTable _custTable = null)

Record reference naming conventions
Use the same naming conventions that are used for entity references.

Id references
In addition to having an entity or record reference, you can introduce the Id reference property.

public CustAccount customerId(CustAccount _custTable = null)

Don't introduce Id references unless you also introduce corresponding entity or buffer references. Id references are shortcuts to the entity or buffer reference methods. The implementation of Id references should delegate the call to the entity or buffer reference.

Id reference naming conventions
Use the UI name if it includes terms such as Id, Number, Account, Code, or Name. Otherwise, add an appropriate suffix to the name of the entity or record reference.

Id reference methods contract
The Id reference method always uses the provided Id to find the referenced entity, and it delegates the call to the entity or record reference method. If no entity or record is found based on the specified Id value, an error message is thrown.
Fluent setter methods

Create fluent setter methods to support the fluent initialization and modification of entities.

Declaration example

```java
public AtlEntitySalesLine setQty(SalesQty _qty)
```

Code example

```java
salesLine.setItem(batchItem).setInventDims([warehouse]).setQty(10).save();
```

Naming convention

```
set<PropertyName>
```

In this naming convention, `<PropertyName>` should match what is used in the name of the corresponding property method.

Action methods

Entities represent not only data but also relevant actions. Actions can be implemented either as a simple action method or as a command object initializer.

Simple action methods

Simple action methods represent a complete action. They should not be fluently chained. The exception is the `save` method, which should be fluent.

Naming convention

```
<ExecuteBusinessOperation>
```

In this naming convention, `<ExecuteBusinessOperation>` is a verb that represents the business operation. It should be the same term that is used on the menu item in the UI.

Examples

```java
salesOrder.save();

salesOrder.postInvoice();
```

Command object initializers

Command object initializers return a command object that lets you specify parameters of the command and run it.

```java
transferLine.pick().setQty(10).setWMSlocation(bulkLocation).execute();
```

Naming convention

```
<ExecuteBusinessOperation>
```

In this naming convention, `<ExecuteBusinessOperation>` is a verb that represents the business operation. It should be the same term that is used on the menu item in the UI.

Examples
salesOrder.pick().execute();
purchaseOrder.register().execute();

**Action entities**

Some actions that are available for an entity can be considered entities themselves. Vendor invoices are one example. Before you post an invoice, you might want to set up parameters of the invoice, edit lines, and save the invoice for later. For these commands, you can introduce a separate entity class.

**Naming convention**

```
new<ActionName>
```

In this naming convention, `<ActionName>` is a noun that represents the business operation. The name should be the UI name of the business operation.

**Example**

```
receipt = transfer.newReceipt().setEditLines(true).setExplodeLines(true);
receipt.lines().withBatch(batch1).single().setReceiptQty(6).setScrapQty(1).save();
receipt.lines().withBatch(batch2).single().setReceiptQty(4).setScrapQty(1).save();
receipt.post();
```

**Class naming convention**

```
AtlEntity<ModuleName><EntityName><ActionName>
```

**Example**

```
AtlEntityInventTransferOrderReceipt
```

**Adding components**

Composition is a relationship where the composite entity has sole responsibility for the disposition of the component parts. The relationship between the composite and the component is a strong “has a” relationship, because the composite object takes ownership of the component. Therefore, the composite is responsible for creating and destroying the component parts.

An object instance can be part of only one composite. If the composite object is destroyed, all the component parts must be destroyed. The component parts have no independent existence outside the composite object, and they can’t be transferred to another object. Composition enforces encapsulation, because the component parts are usually members of the composite object.

An example of a composite object is a source document that is made up of source document lines.

**Example**

In the source document example, the document entity serves as the composition root and is responsible for creating any new instances of document lines. In this case, the source document entity will have an `addLine()` method that initializes and returns a new line for the document.

```
public AtlEntitySalesLine addLine()
```

The `addLine()` method adds the line object (`salesLine` in this example) to a collection of lines and returns the parent entity (`salesOrder` in this example) to preserve the fluency of application programming interfaces (APIs). To create a new line, create a `newLine()` method.

**Naming convention for adding components**
Methods for adding components should use the UI names of the buttons.

**Example**

```java
salesLine = salesOrder.addLine();
```

**Component collections**

You can search for components by using query methods.

**Naming convention for component collections**

Methods for accessing component collections should use the UI names of the grid on the hosting page.

**Query methods**

Query methods on an entity let you search for components and related entities.

**Example**

```java
transferOrderLine = transferOrder.lines().withItem(item).single();
```

In this example, `lines()` is a query method that returns the `AtlQueryTransferOrderLines` query. This query is already filtered so that it returns only transfer order lines for the transfer order that the `lines()` method was called on.

**Naming convention**

Use the UI names whenever you can. Abbreviations are acceptable if the UI name is too long to be used in test automation.

**Example**

```java
public AtlQueryWHSLoadLines lines()
{
    return new AtlQueryWHSLoadLines().forLoadId(this.parmLoadId());
}
```

**Additional resources**

Queries in the Acceptance test library
Command classes are responsible for running business operations. They let you use fluent application programming interfaces (APIs) to set the parameters of these operations.

### Naming convention

#### Command class name

`AtlCommand<ModuleName><EntityName><ExecuteBusinessOperation>`

In this naming convention:

- `<ModuleName>` is based on the names of the modules on the main menu. You should use a short version or an abbreviation to support brevity of test code.
- `<EntityName>` is optional and is used when the command applies to different types of entities.
- `<ExecuteBusinessOperation>` is a verb that represents the name of the business operation.

#### Examples

- `AtlCommandInventMark`
- `AtlCommandSalesReturnOrderLineRegister`

### Implementation

Command objects that are returned should implement the `AtlICommand` interface and should inherit from the `AtlCommand` class.

Command objects should provide fluent setter methods that are used to set the parameters of the command.

#### Example

```java
salesLine.pick().setInventDims([locationOut]).setQty(pickedQty).execute();
```

### Fluent setter methods

Commands allow for two types of fluent setter methods, `for` methods and `set` methods:

- **for** – These methods are used for command parameters that represent the entities that the command applies to.

  For example, the invoice command can apply to sales orders. Therefore, a `for` method is used to set the sales order that the invoice command applies to.

- **set** – These methods are used for all other parameters of the command.

  For example, when you reserve a sales line, you usually specify a quantity. The quantity isn't something that the command applies to but is instead a simple parameter of the command. Therefore, a `set` method is used to specify the quantity parameter of the reservation command.
In this naming convention, `<CommandParameterName>` is the name of the parameter that is being set for the command by using the fluent method.

**Examples**

```java
onHandAdjustment.forItem(item).setQuantity(10).execute();
picking.forSalesLine(salesLine).setInventDims([warehouse, batch1]).setQuantity(10).execute();
```
Creator classes provide fluent application programming interfaces (APIs) that are used to create test data.

**Naming convention**

`AtlCreator<ModuleName><EntityName>`

In this naming convention:
- `<ModuleName>` is optional and is based on the names of the modules on the main menu. You should use a short version or an abbreviation to support brevity of test code.
- `<EntityName>` is optional and is used when the command applies to different types of entities.

**Examples**

- `AtlCreatorCostGroup`
- `AtlCreatorCustomer`

**Fluent setter methods**

Creator classes should provide fluent setter methods that are used to set the properties of the entity that is being constructed.

**Naming convention**

`set<EntityPropertyName>`

In this naming convention, `<EntityPropertyName>` is the name of the property that is being set for the entity by using the fluent method.

**Example**

```java
item = new AtlCreatorProductsReleasedVariant()
    .setItemId('DemoItem')
    .setColor(ecoResColor)
    .getStyle(ecoResStyle)
    .setConfig(ecoResConfig)
    .setSize(ecoResSize)
    .create();
```

**When should creators be used instead of entities?**

For information that will help you choose between entities and creators, see Should I implement an entity or a creator class.
A query class provides fluent application programming interfaces (APIs) that are used to find an instance of the corresponding entity, based on various criteria. Query classes are often used in validation scenarios. They are usually used together with specifications.

**Naming convention**

```
AtlQuery<ModuleName><EntityNamePlural>
```

In this naming convention:
- `<ModuleName>` is optional and is based on the names of the modules on the main menu. However, a short version or an abbreviation should be used to support brevity of test code.
- `<EntityNamePlural>` is the plural version of the entity name.

**Examples**

- `AtlQueryWHSLoadLines`
- `AtlQueryInventTransferOrderLines`

**Implementation**

Query classes inherit from the `AtlQuery` class that is common to all queries.

**Fluent setters**

Query classes should provide fluent setter methods to specify ranges for the query.

**Example**

```
loadLine = data.whs().loadLines().query().forSalesOrder(salesOrder).single();
```

Queries allow for two types of fluent setter methods, `for` methods and `with` methods:
- **for** – These methods are used for filters of the query that act as parents of composition or aggregation relationships. For example, the sales lines query exposes a `for` method to filter sales lines for a specific sales order.
- **with** – These methods are used for all other ranges of the query.

**Naming convention**

```
for<QueryRangeName>
```

```
with<QueryRangeName>
```

In this naming convention, `<QueryRangeName>` is the name of the field that the range is applied on.

**Examples**
loadLine = data.whs().loadLines().query().forLoad(load).withInventQty(10).single();

transferLine =
data.invent().transferOrderLines().query().forTransferOrder(transferOrder).withInventDims([batch1]).single();
A specification class provides fluent application programming interfaces (APIs) that are used to define the set of criteria that an entity should meet. Specifications are often used in validation scenarios. They are usually used together with query classes.

An advantage of specification classes is that the validation code becomes very concise and expressive. Basically, you can do multiple validations in a single line of code.

**Naming convention**

\[
\text{AtlSpec<ModuleName><#EntityName>}
\]

In this naming convention:

- `<ModuleName>` is optional and is based on the names of the modules on the main menu. However, a short version or an abbreviation should be used to support brevity of test code.
- `<#EntityName>` represents the name of the entity that is used throughout the Acceptance test library (ATL).

**Examples**

- `AtlSpecWHSLoadLine`
- `AtlSpecWHSWorkLine`

**Implementation**

Specification classes should provide fluent setter methods to specify various criteria of the specification.

**Example**

The following code verifies that the work contains six lines that meet the specified criteria. For example, the first line should have 1 as the line number of 1, Pick as the work type, 1 as the quantity, Closed as the status, and bulk as the location.

```javascript
work.lines().assertExpectedLines(
    worklines.spec().withLineNum(1).withWorkType(WHSWorkType::Pick).setQuantity(1)
        .setStatus(WHSWorkStatus::Closed).setLocation(locations.bulk()),
    worklines.spec().withLineNum(2).withWorkType(WHSWorkType::Pick).setQuantity(1)
        .setStatus(WHSWorkStatus::Closed).setLocation(locations.floor()),
    worklines.spec().withLineNum(3).withWorkType(WHSWorkType::Put)
        .setQuantity(2).setStatus(WHSWorkStatus::Closed).setLocation(locations.stage()),
    worklines.spec().withLineNum(4).withWorkType(WHSWorkType::Pick).setQuantity(2)
        .setStatus(WHSWorkStatus::Cancelled).setLocation(locations.stage()),
    worklines.spec().withLineNum(5).withWorkType(WHSWorkType::Put).setQuantity(2).setStatus(WHSWorkStatus::Cancelled)
);
```
The Acceptance test library (ATL) code generator quickly generates and updates new ATL entities, queries, and specifications, based on tables and data entities.

Create the AtlEntity class by using the wizard

Follow these steps to create the **AtlEntity** class by using the **Code generation** wizard.

1. In Microsoft Visual Studio, open the table in the designer window.
2. Right-click the name of the table, and then, on the **Add-ins** menu, select **Generate ATL Entity**.
3. Select the fields that should be included in the **AtlEntity** class, and then select **Add**.
4. Rename the entity and the fields as you require.
5. Select **Generate** to create the class.

Additional optional steps

When you create the **AtlEntity** class, you can also complete these tasks:

- Add required actions for the scenario.
- Add a **default** method to **AtlData** classes.
- Override the **setMainRecordField** method to call the **modifiedField(_fieldId)** method on the table.

```java
protected void setMainRecordField(FieldId _fieldId, anytype _value) {
    super(_fieldId, _value);
    common.modifiedField(_fieldId);
}
```

Create the AtlQuery class by using the wizard

Follow these steps to create the **AtlQuery** class by using the **Code generation** wizard.

1. In Visual Studio, open the table in the designer window.
2. Right-click the name of the table, and then, on the **Add-ins** menu, select **Generate ATL Query**.
3. Select the fields and relations that should be included in the **AtlQuery** class, and then select **Add**.
4. Rename the query, the fields, and the relations as you require.
5. Select **Generate** to create the class.

Additional optional steps

When you create the **AtlQuery** class, you can also add a **query** method to the **AtlData** class that returns an instance of the **AtlQuery** class that you created earlier in this topic.

Create the AtlSpec class by using the wizard

Follow these steps to create the **AtlSpec** class by using the **Code generation** wizard.

1. In Visual Studio, open the table in the designer window.
2. Right-click the name of the table, and then, on the **Add-ins** menu, select **Generate ATL Specification**.
3. Select the fields that should be included in the `AtlSpec` class, and then select **Add**.

4. Rename the specification and the fields as you require.

5. Select **Generate** to create the class.

Additional optional steps

Add a `spec` method to the data class that returns an instance of the `AtlSpec` class that you created earlier in this topic.
Use `var` and declare variables inline

- Use the `var` keyword (type inference).
- Declare variables inline instead of in a separate statement.

**Do this**

```javascript
var item = items.default();
var salesOrder = data.sales().salesOrders().createDefault();
var salesLine = salesOrder.addLine().setItem(item).setInventDims([warehouse]).setQuantity(10).save();
```

**Don’t do this**

```javascript
InventTable item;
AtlEntitySalesOrder salesOrder;
AtlEntitySaleOrderLine salesLine;

--
item = items.default();
salesOrder = data.sales().salesOrders().createDefault();
salesLine = salesOrder.addLine().setItem(item).setInventDims([warehouse]).setQuantity(10).save();
```

**Justification**

The advantages of using `var` are that you write less code, you don’t have to remember exact type names, and the test logic isn’t cluttered with unimportant information. Overall, the test code easier to read.

In the previous example, it doesn’t matter whether `item` is of the `ItemId`, `InventTable`, or `AtlEntityInventItem` type. The important detail is that you’re creating a sales line that has a well-known default item. The exact types of the `salesOrder` and `salesLine` variables aren’t important. The contracts of these types are clear from the naming and usage.

**Considerations**

- Don’t use type inference if you want compilation to fail if the return type of a method changes.
- Don’t use type inference if you can’t invent meaningful variable or method names.

Use entities instead of IDs as method parameters

Well-known data methods, creator methods, and `init` methods usually return records or entities instead of IDs. We recommend that you use records or entities as method parameters.

**Do this**

```javascript
var salesLine = salesOrder.addLine().setItem(item).save();
```

**Don’t do this**

```javascript
var salesLine = salesOrder.addLine().setItemId(item.ItemId).save();
```
**Justification**
The code is easier to read, because it isn't cluttered with unimportant technicalities.

**Considerations**
If you know only the ID, use the method that takes the ID as an argument.

**Use navigation node shortcuts**
When you automate a new domain area, introduce a base class that holds shortcuts to the most frequently used navigation objects in that area.

For example, for the warehouse management area, there is a base class that is named `AtlWHSTestCase`. It contains shortcuts to `data.whs()`, `data.invent()`, `data.invent().items()`, `data.invent().units()`, and other navigation objects. The shortcuts simplify your test code.

```java
class AtlWHSTestCase extends SysTestCase {
    AtlDataRootNote data;
    AtlDataInvent invent;
    AtlDataInventOnHand onHand;
    AtlDataProductItems items;
    AtlDataWHS whs;

    protected void initDataSetupReferences() {
        data = new AtlDataRootNode();
        invent = data.invent();
        onHand = data.invent().onHand();
        items = data.invent().items();
        whs = data.whs();
    }
}
```

It also makes sense to introduce shortcuts that are shared among many test methods in the same class.

**Do this**

```java
class WHSMinMaxReplenishmentScenarioTest extends AtlWHSTestCase {
    var item = items.default();
    var warehouse = invent.warehouses().default();
}
```

**Don't do this**

```java
class WHSMinMaxReplenishmentScenarioTest extends SysTestCase {
    var item = data.invent().items().default();
    var warehouse = data.invent().warehouses().default();
}
```

**Considerations**
You don’t have to create a shortcut for every navigation node that you need. However, consider creating them for the navigation nodes that are frequently used.
Which fluent prefix should I use: set, for, or with?

Depending on the class that you want to add a fluent method to, different rules might apply:

- Entities
- Creators
- Commands
- Queries
- Specifications

Should I implement an entity or a creator class?

For most entities, the effort of creating an entity class is the same as the effort of creating a creator class. Therefore, the entity class should be created. However, in some cases, the process of creating an entity might not be straightforward. A good example is the Item entity. Because more than ten different tables make up the Item entity, it's hard to create the entity class. Because you will almost never have to update existing items in your test cases, it's OK if you just have a creator class, which is much easier to implement.

Does the order of the chained fluent setters matter?

In most of the cases, the order of the chained fluent setters doesn't matter. However, be aware that defaulting occurs at the time of the call to the setter method. Therefore, a change in the order of the methods can produce different results. Here is an example for the sales line.

**Option 1**

```java
salesLine.setQuantity(10).setUnitPrice(100).setAmount(2000).save()
```

This option produces a sales line where the amount == 2,000, because the amount is set last.

**Option 2**

```java
```

This option produces a sales line where the amount == 1,000, because after you set the unit price, the amount is set to quantity × price by default.

Can I use ATL for tests that run on the empty data set?

The Acceptance test library (ATL) can be used on the empty data set without issues. The automatic setup of prerequisites is done on demand. For example, prerequisites for invoice posting will be set up only during the first call to `salesOrder.postInvoice()`. For more information, see Ensure.

Ensure methods can also be called in the `setUpTestCase` method to improve performance if more than one test in your test class must post invoices.
Can I use ATL for unit tests?

ATL should be used mostly for data setup and validation in integration and component tests. However, in some cases, it's also used for unit tests.

Why should I be cautious about using ATL in unit and component tests?

In some of the more complex entities, such as Sales order, all the business logic that is associated with the modifiedField, insert, and update events is called. Creation of invoice transactions, for example, is also done by running real invoice posting logic. Therefore, the performance of some operations will be slow. However, these issues don't occur for most of the entities that represent master data. Therefore, you should be able to use those entities in any type of test.

There should not be significant overhead if specifications and queries are used to do validation. These artifacts can also be used in unit tests.
This topic provides information about date-effective data entities and data sources, and shows how to create a date-effective entity. It also explains how date effectivity applies to read and write activities.

There are different design patterns for date-effective features that involve data entities. The patterns are classified into two main categories:

- **Date-effective entities** – The entity has at least one date-effective data source, and the entity itself is also date effective.
- **Non-date-effective entities** – The entity itself is not date effective, but it does contain date-effective data sources.

The next sections describe the small list of properties and methods that control the date-effective behavior of entities and their date-effective data sources.

## Date-effective entities

The following table describes the properties that control the date-effective behavior of a data entity.

<table>
<thead>
<tr>
<th>PROPERTY NAME OF THE ENTITY</th>
<th>NODE OF THE PROPERTY</th>
<th>VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ValidTimeStateEnabled</td>
<td>Data entity node in the designer</td>
<td>Yes (or No)</td>
<td>The value <strong>Yes</strong> makes the entity date effective. The entity must have <strong>ValidFrom</strong> and <strong>ValidTo</strong> fields. These fields are mapped to the <strong>ValidFrom</strong> and <strong>ValidTo</strong> fields of a date-effective data source. The value <strong>No</strong> does <em>not</em> disable the enforcement of date effectivity on any date-effective tables that are data sources of the entity.</td>
</tr>
<tr>
<td>ValidTimeStateKey</td>
<td>Under the data entity node, <strong>Keys &gt; EntityKey</strong></td>
<td>Yes (or No)</td>
<td>The value <strong>Yes</strong> identifies the key that is required to enforce the date-effective values on this particular entity.</td>
</tr>
</tbody>
</table>

### Read activities

When date effectivity is set at the data entity level, reads from the entity behave the same way as reads from a table. The entity has **ValidFrom** and **ValidTo** fields that the system applies date filters to during reads.

#### Query modes and the validtimestate keyword of X++ SQL select

A date-effective entity supports the following three *query modes*, which vary in their use of the X++ `validtimestate` keyword:

- **Default mode** – Current records are returned using `select * from FMVehicleRateEntity; // X++ SQL`. 


Applying a date filter at the data source level

There are scenarios where date-effective filtering is required outside the data entity, at the data source level. For example, the customer entity (CustTableTestEntity) contains CustTable and LogisticsPostalAddress as data sources, where LogisticsPostalAddress is a date-effective table and CustTable is a regular table. The purpose of a customer entity is to have a list of customers and their active primary addresses, if they have primary addresses. Therefore, the customer entity itself isn't date effective, but it requires date filters on one of the data sources. In this case, the entity isn't marked `ValidTimeStateEnabled`. Instead, an **Apply Date Filter** property is added on the data source. If the value of **Apply Date Filter** is set to **Yes**, date filters are automatically applied to that data source. The following table describes the properties that control the date-effective behavior of a date-effective data source of a data entity.

<table>
<thead>
<tr>
<th>PROPERTY NAME OF THE DATA SOURCE</th>
<th>NODE OF THE PROPERTY</th>
<th>VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply Date Filter</td>
<td>Node of any particular data source of the entity</td>
<td>Yes (or No)</td>
<td>For reads, this property controls whether date filters are applied on the entity data source. In this case, the data source should be marked <code>ValidTimeStateEnabled</code>. This property value has effect regardless of whether the entity itself is date effective. For writes, this property has no effect.</td>
</tr>
</tbody>
</table>

State matrixes for reads

This section concerns only reads from the data entity. The following pair of reference matrixes describe the combinations of date-effective states that can exist between a data entity and its data source. Each table contains four cases, and each case discusses two distinct targets. Here are the primary points that you should understand:

- On any given read from the entity, the query mode is the same for both the entity and date-effective data sources.
- If the entity is not date effective, the query mode is limited to the default mode. Therefore, the date-effective data source is accessed only for the current date.
- On the date-effective data source, the **Apply Date Filter** property can be set to **No** to make the data source return all data – past, current, and future.
- For OData, date-effective filters are not applied to the data entity. However, filters on the data source are applied at all code paths.

A. **Entity is date effective, because ValidTimeStateEnabled = Yes**

**Data source is date effective**

**Data source is not date effective**

---

- **AsOfDate mode** – Records valid for the specified date are returned using
  ```sql
  select validtimestate(d1) * from FMVehicleRateEntity;
  ```
- **AsOfDateRange mode** – Records valid for the specified date range are returned using
  ```sql
  select validtimestate(d1,d2) * from FMVehicleRateEntity;
  ```

**Important:** For data entities that aren’t themselves date effective, but that have a data-effective data source, only the default query mode is available. This concept is discussed later in this article.
Apply Date Filter = Yes

- **Entity**: Date filters are applied. Any query mode is supported.
- **Data source**: Filters are applied. Any query mode is supported, but the mode is the same as is coded for the entity.

Non-date-effective data sources aren't affected.

Apply Date Filter = No

- **Entity**: Date filters are applied. Any query mode is supported.
- **Data source**: No date filters are applied.

Non-date-effective data sources aren't affected.

B. Entity is *not* date effective, because ValidTimeStateEnabled = No

Data source *is* date effective

Data source *is not* date effective

Apply Date Filter = Yes

- **Entity**: No date filters are applied.
- **Data source**: Date filters are applied. Only the default query mode is supported, where the X++ `validtimestate` keyword is omitted.

Non-date-effective data sources aren't affected.

Apply Date Filter = No

- **Entity**: No date filters are applied.
- **Data source**: No date filters are applied.

Non-date-effective data sources aren't affected.

The following screen shot shows the **Apply Date Filter** property set to **Yes**. Therefore, date filters will be applied to reads of the **Address** data source.
Write activities

This section describes your options for configuring the behavior of date-effective entities and their date-effective data sources. We will start by reviewing the concept of date-effective tables and contrasting them with date-effective entities. **Date-effective table:** When data is inserted or updated in a date-effective table, the process has the option of calling the `xRecord.validTimeStateUpdateMode` method on the table buffer. The method accepts an element of the `ValidTimeStateUpdate` enumeration. Here are the available element values:

- CreateNewTimePeriod
- Correction
- EffectiveBased

**Date-effective entity:** By contrast, when data is inserted or updated in a date-effective data entity, the `validTimeStateUpdateMode` method isn’t used at the entity level. For writes, the data entity leaves the date-effective processing to the table level. You can use the `Valid Time State Update` property on the entity data source to specify the `validTimeStateUpdateMode` method to use for each data source of the data entity.

Creating a date-effective entity

This section shows how to create a date-effective entity.

**Create a new project**

1. Click File > New > Project to create a new project.

2. In Solution Explorer, right-click your project, and then click Properties. The Property Pages dialog box for your project opens.

3. Change the value of the `Synchronize database on build` property to True, and then click OK. You must set this property only one time per project.
Add a new data entity to your project

Create a new entity that is named **FMVehicleRateEntity**, and add it to the project.

1. In the left pane, select **Microsoft Dynamics 365 Artifacts**, and then click **Data Entity** in the left column of the main pane.

2. Click **Add**. The **Data Entity View** wizard starts.

3. Specify the property values for the data entity that you are creating, as shown in the following screenshot. The most important field is **Primary data source**, where you select **FMVehicleRate**.

4. Add fields to the entity from the primary data source, **FMVehicleRate**.

5. Select all fields, and then click **Finish**.
The items are added to the project in Solution Explorer.

Build your project
1. Click Build > Build Solution to build your project.
2. Verify that the build has no errors. Warnings should be tolerated at this stage in the process.

Validate the property values
- In Solution Explorer, select the FMVehicleRateEntity node, and validate the properties of the FMVehicleRateEntity entity by comparing them to the values in the Properties pane.

Make your entity date effective
1. In Solution Explorer, right-click the FMVehicleRateEntity node, and then click Open. The designer for the entity opens in the middle pane.

2. Change the value of the Validate Time State Enabled property to Yes.
/// <summary>
/// Runs the class with the specified arguments.
/// </summary>
/// <param name = "_args">The specified arguments.</param>
public static void main(Args _args)
{
    FMVehicleRateEntity FMVehicleRateEntity;
    FMCarClass vehicle;
    FMVehicleModel model;
    FMVehicleRate vehicleRateTable;
    TransDate d1=1\1\1999,d2=31\12\2014;

    ttsbegin;

    select count(RecId) from FMVehicleRateEntity;
    info(strfmt("Entity - Valid today before insert %1",FMVehicleRateEntity.RecId));

    select count(RecId) from vehicleRateTable;
    info(strfmt("Table - Valid today before insert %1",vehicleRateTable.RecId));

    select firstonly model;
    vehicle.VehicleModel = model.RecId;
    vehicle.VehicleId = "TestV1001";
    vehicle.insert();

    if (vehicle)
    {
        FMVehicleRateEntity.clear();
        FMVehicleRateEntity.FMVehicle_VehicleId = vehicle.VehicleId;
        FMVehicleRateEntity.ValidFrom = d1;
        FMVehicleRateEntity.ValidTo = d2;
        FMVehicleRateEntity.RatePerDay = 100;
        FMVehicleRateEntity.RatePerWeek = 600;
        FMVehicleRateEntity.insert();

        // Should increase by one as compared to before insert numbers
        select count(RecId) from FMVehicleRateEntity;
        info(strfmt("Entity - Valid today after insert %1",FMVehicleRateEntity.RecId));

        // Should increase by one as compared to before insert numbers
        select count(RecId) from vehicleRateTable;
    }

    Build your project again, and run the following X++ code to test your project.
select count(RecId) from vehicleRateTable;
info(strfmt("Table - Valid today after insert %1",vehicleRateTable.RecId));

// New record should show in count
select validtimestate(d1) count(RecId) from FMVehicleRateEntity;
info(strfmt("Entity - Valid 1999 %1",FMVehicleRateEntity.RecId));

// New record should show in count
select validtimestate(d1) count(RecId) from vehicleRateTable;
info(strfmt("Table - Valid 1999 %1",vehicleRateTable.RecId));

// update newly created record
// This should split record into two - 2009 to Today, today to 2014
// Split happens because of mode in saveEntityDatasource
select forupdate validtimestate(d1,d2) FMVehicleRateEntity
where FMVehicleRateEntity.FMVehicle_VehicleId == vehicle.VehicleId &&
    FMVehicleRateEntity.ValidFrom == d1 &&
    FMVehicleRateEntity.ValidTo == d2;
FMVehicleRateEntity.RatePerDay = 200;
FMVehicleRateEntity.update();

// validate the split
while select validtimestate(d1,d2) FMVehicleRateEntity
    where FMVehicleRateEntity.FMVehicle_VehicleId == vehicle.VehicleId
{
    info(strfmt("Entity - %1 to %2 , RatePerDay-%3, RatePerWeek-%4",
                FMVehicleRateEntity.ValidFrom,
                FMVehicleRateEntity.ValidTo,
                FMVehicleRateEntity.RatePerDay,
                FMVehicleRateEntity.RatePerWeek));
}

ttsabort;
}
This topic provides links to topics about development by independent software vendors (ISVs).

- Link X++ modules to packages by using ISV Studio
- ISV licensing
- ISV licensing on-premises
Independent software vendors (ISVs) can link their X++ modules to their registered products and solutions by using Microsoft Power Platform ISV Studio. Linking enables ISV’s to monitor the success and usage of their applications in Finance and Operations apps.

**NOTE**
For the link from X++ into ISV Studio to work correctly, customers need to have deployed ISV packages with the correct solution ID in all the ISV models. The customer's environment also has to be version 10.0.16 or higher.

### Find the product ID in Microsoft Partner Center

Sign in to Partner Center and open the Offer overview page for your product. From the browser’s URL bar, locate the product ID globally unique identifier (GUID), as shown in the following example.


**NOTE**
The product ID does not necessarily match the offer code of your product, although they may be similar. Using the offer code in your descriptors will not correctly identify your X++ modules to ISV Studio.

### Update your X++ model descriptors

For all models that make up your solution, locate the descriptor XML files. For every descriptor that belongs to a solution, update the SolutionId tag with the product ID from Partner Center. The order of the elements must match the following example to get the expected results.
After you recompile, the X++ binaries will contain the product ID and will link to ISV Studio after they are deployed to a Tier 2+ sandbox or production environment.
This topic describes the independent software vendor (ISV) licensing feature. It includes information about benefits and capabilities of the ISV licensing feature, and explains how to enable licensing for an ISV solution, create a package and generate a customer-specific license, and create self-signed certificates for test purposes.

The Microsoft Dynamics ecosystem provides tools and frameworks that let independent software vendors (ISVs) build, deploy, sell, and therefore monetize vertical industry solutions that can be repackaged. The ISV licensing feature provides the following benefits:

- It provides a safer licensing mechanism for ISV solutions for customers and partners. ISV solutions are enabled only if the customer has purchased a valid license key from the ISV.
- It aligns how customers handle licenses for ISV solutions from different ISVs, and therefore lowers the total cost of ownership (TCO).
- ISVs can independently generate, manage, and distribute ISV licenses by using industry standard frameworks.

This feature doesn’t enable ISV competitor copycat protection (that is, source-based protection).

Capabilities

This section describes various capabilities of the ISV licensing feature.

**ISVs can generate their own licenses**

ISVs can independently generate their own licenses, apply them to solutions, and deliver those solutions to partners and customers. Each ISV license enables run-time features that help protect the ISV solution. Additionally, each ISV license is tied to an ISV Authenticode certificate, which ensures that the software was distributed by the ISV.

A run-time check makes sure that an ISV-generated license key exists in the customer's environment

Each ISV solution that is tied to a license runs only when a valid license key exists in the customer’s environment. Therefore, if an ISV ties its solution to a license, but the customer doesn’t have a valid license key, the solution doesn’t run.

**There are two types of license: Boolean and Number**

ISVs can create two types of license: **Boolean** and **Number**. ISVs can associate an expiration date with either type of license. This expiration date is applied only to the ISV licenses and is independent of the system expiration date. A Boolean license is a simple activation license. The type of license (**Boolean** or **Number**) is set through a property in the license code node. ISVs can write their own custom logic to check the count that is provided in the ISV license, to make sure that their solutions are being used within the license terms. For more information, see Licensing Framework for ISVs.

License validation errors

When an ISV license becomes invalid after import, the ISV solution continues to run until the server is restarted. (After the server is restarted, the solution is disabled.) An error is thrown when the instance of the Application Object Server (AOS) starts. The error is written to the event log.

Implementing ISV licensing in a solution

ISVs must have a valid Authenticode certificate (X.509) from a certificate authority (CA). Microsoft doesn't
Certificate import and export

The certificate is used to sign your customer license files and validate the license files at the time of import. Authenticode certificates support four file formats. For the ISV licensing feature, you must have the certificate files in two formats:

- **Personal Information Exchange (PFX, also known as PKCS #12)** – The PKCS #12 format, which uses the .pfx file name extension, supports secure storage of certificates, private keys, and all certificates in a certification path. The PKCS #12 format is the only file format that can be used to export a certificate and its private key.
- **Base64-encoded X.509** – The Base64 format supports storage of a single certificate. This format doesn’t support storage of the private key or certification path.

There is a restriction on the format. The PFX (PKCS #12) format should be used only to export the certificate together with its private key for signing/generating purposes. It should never be shared outside the ISV organization. The DER-encoded binary X.509 format, which uses the .cer file name extension, should be used to export the public key of the certificate that must be embedded in the Application Object Tree (AOT) License. This public key is distributed to customers via the model. It’s used when a license is imported, to make sure that the license is signed by the ISV license that owns the private key.

Enable licensing for your ISV solution

Follow these steps to enable licensing for your solution.

1. Create an ISV solution. In Visual Studio, click **File > New project**. In the **New Project** dialog, click **Installed > Templates > Dynamics 365**. Create a **Finance Operations** project. In this example, we named the project **NewISVSolution**.
2. Add the certificate’s public key (.cer file) to your project as a resource. To create a certificate for testing, see Appendix: Create self-signed certificates for test purposes.

   a. Right-click the project in Solution Explorer, then click **Add > New item**.

   b. Under **Installed > Dynamics 365 Items**, click **Labels And Resources**, and then select **Resource**. Name the resource. In this example, we named the resource **ISVCert**.

   c. Click **Add** and select the certificate’s public key file (.cer file).
d. Click **Open** to add the certificate.

3. Create a license code. Right-click the project in Solution Explorer, then click **Add** > **New item**. Under **Installed** > **Dynamics 365 Items**, choose **Configuration**. In the list, choose **License Code** and name the license code. In this example, we named the license code **ISVLicenseCode**. Click **Add**.

4. Map the certificate to the license code. In the Properties window for the license code, set the **Certificate** property to your certificate resource. In this example, we set **Certificate** to **ISVCert**.
5. Create one or more configuration keys. Right-click the project in Solution Explorer, then click **Add > New item**. Under **Installed > Dynamics 365 Items**, choose **Configuration**. In the list, choose **Configuration Key**. Name the key and click **Add**. In this example, we named the configuration key **ISVConfigurationKey1**.

6. Associate the license code with the configuration key. In Solution Explorer, double-click the configuration key to open the Properties window. In the Properties window, set the **LicenseCode** property to your license code. In this example, we set the **LicenseCode** to **ISVLicenseCode**.

7. Associate a configuration key to an element in your solution. For example, create a new form. Right-click the project in Solution Explorer, then click **Add > New item**. Under **Installed > Dynamics 365 Items**, choose **User Interface**. In the list, choose **Form** and give it a name. In this example, we named the form **ISVForm**.
8. Add a button to the form. Double-click the form in the Solution Explorer. In the Design window, right-click and select **New**, and then **Button**. Set the **Text** property to **ISVButton**.

At runtime, the button is visible because it isn’t controlled by a configuration key at first.

9. Associate a configuration key with the button. In the Properties window for the button, set the **Configuration Key** property to your configuration. In this example, we set the **Configuration Key** to **ISVConfigurationKey1**.

At runtime, the button is not visible because the configuration key must be available and enabled.
Create a package and generate a customer-specific license

1. Collect the tenant name and ID for the customer to issue the license to. You can find this information at Settings > Help & Support > About on the Licenses tab.

2. Generate a license for the customer (tenant ID and name), and sign the license by using the certificate's private key. You must pass the following parameters to the `axutil genlicense` command to create the license file.

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>The name of your license file.</td>
</tr>
<tr>
<td>licensecode</td>
<td>The name of your license code (from Microsoft Visual Studio).</td>
</tr>
<tr>
<td>certificatepath</td>
<td>The path of your certificate's private key.</td>
</tr>
<tr>
<td>password</td>
<td>The password for your certificate's private key.</td>
</tr>
<tr>
<td>customer</td>
<td>The customer's tenant name (from the screenshot under step 1).</td>
</tr>
</tbody>
</table>
The customer’s tenant ID (labeled “Serial number” in the screenshot).

Optional: The expiration date for the license.

Optional: The number that custom validation logic can use as required. This could be users, but is not limited to users.

Here is an example.

C:\AOSService\PackagesLocalDirectory\Bin\axutil genlicense /file:c:\templicense.txt /certificatepath:c:\tempsvcert.pfx /licensecode:ISVLicenseCode /customer:TAEOfficial.csctp.net /serialnumber:4dbfcf74-c5a6-4727-b638-d56e511df381 /password:*******

3. Import the license into the target environment.

NOTE
In production systems, you complete this step from Microsoft Dynamics Lifecycle Services (LCS), by using a deployable package. For more information, see the “Production environments” section later in this topic.

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>--setupmode importlicensefile</td>
<td>Use this parameter to inform the setup tool that a license will be loaded.</td>
</tr>
<tr>
<td>--metadatadir</td>
<td>Use this parameter to specify the metadata directory. You should use the default packages directory.</td>
</tr>
<tr>
<td>--bindir</td>
<td>Use this parameter to specify the binaries directory. You should use the default packages directory.</td>
</tr>
<tr>
<td>--sqlserver</td>
<td>Use this parameter to specify the Microsoft SQL Server. For one-box environment, use a period (.).</td>
</tr>
<tr>
<td>--sqldatabase</td>
<td>Use this parameter to specify the SQL Server database. For one-box environments, use AXDB.</td>
</tr>
<tr>
<td>--sqluser</td>
<td>Use this parameter to specify the SQL Server user. You should use axdbadmin.</td>
</tr>
<tr>
<td>--sqlpwd</td>
<td>Use this parameter to specify the SQL Server password.</td>
</tr>
<tr>
<td>--licensefilename</td>
<td>Use this parameter to specify the license file that will be loaded.</td>
</tr>
</tbody>
</table>

Here is an example.
4. The corresponding configuration key will be available and enabled on the License configuration page. By default, the configuration is enabled. For example, see the ISVConfigurationKey1 configuration key in the following screenshot.

5. In non-production installations, you must start the database synchronization process from Visual Studio. After the configuration key is enabled, the button becomes visible, as shown in the following screenshot.

Protection best practices

Solutions can be delivered in two forms:

- Model files (source code)
- Deployable packages (binary)

To protect your configuration keys and license codes, we recommend that you release them in binary form, by using a deployable package. Customers will then be able to install and interact with those elements in Visual Studio. Although customers will be able to refer to items in the deployable package, they won't be able to access source code or make modifications to the items. (However, they can create extensions.) More details about the capability to release solutions in binary form will be available soon. The deployable package (binary) can also include classes and other logic that your customer doesn't require access to and should not be able to customize.
To install ISV licenses in production systems, you must use a deployable package through LCS. You can find a template package for configuration mode at the following location in all installations:

\<PackagesFolder>\bin\CustomDeployablePackage\ImportISVLicense.zip

(Packages folder is typically under \j\AOSService\PackagesLocalDirectory or \c\AOSService\PackagesLocalDirectory\)

1. Make a copy of the package template.
2. Put the license file in the following folder within the package template:
   \ImportISVLicense.zip\AosService\Scripts\License
Appendix: Create self-signed certificates for test purposes

NOTE
Self-signed certificates can be used only during development. They aren’t supported in production environments.

For Platform update 34 and earlier: (Deprecated - uses SHA1 hash algorithm for license creation)

1. For test purposes, create a self-signed CA certificate. Use the Visual Studio tools prompt to run the following command.

   ```
   makecert -r -pe -n "CN=IsvCertTestAuthority O=IsvCertTestAuthority" -ss CA -sr LocalMachine -a sha256 -len 2048 -cy authority -sky signature -b 01/01/2016 -sv c:\temp\CA.pvk c:\temp\CA.cer
   ```

2. Create a certificate by using the CA.

   ```
   makecert -pe -n "CN=IsvCertTest O=IsvCertTest" -ss ISVStore -sr LocalMachine -a sha256 -len 2048 -cy end -sky signature -eku 1.3.6.1.5.5.7.3.3 -ic c:\temp\ca.cer -iv c:\temp\ca.pvk -b **/**/**** -sv c:\temp\isvcert.pvk c:\temp\isvcert.cer
   ```

3. Convert the ISV certificate to PFX format.

   ```
   pvk2pfx -pvk c:\temp\isvcert.pvk -spc c:\temp\isvcert.cer -pfx c:\temp\isvcert.pfx -po ********
   ```

4. For a test scenario, import the self-signed CA certificate manually on all the AOS instances.

   ```
   certutil -addstore root c:\temp\ca.cer
   ```

   However, if a self-signed ISV certificate was used, that certificate must be imported instead of the CA certificate.

   ```
   certutil -addstore root c:\temp\isvcert.cer
   ```

For Platform update 35 and later: (Uses SHA256 hash algorithm for license creation)

1. For test purposes, create a self-signed certificate using the PowerShell command

   ```
   $cert = New-SelfSignedCertificate -CertStoreLocation Cert:\LocalMachine\My -DnsName "IsvCert" -Type CodeSigningCert -KeyExportPolicy Exportable -HashAlgorithm sha256 -KeyLength 2048 -KeySpec Signature -Provider "Microsoft Enhanced RSA and AES Cryptographic Provider" -NotBefore (Get-Date -Year 2020 -Month 1 -Day 1) -NotAfter (Get-Date -Year 2022 -Month 12 -Day 31)
   ```

   a. Create the certificate. (Note: adjust start and end dates accordingly.)

   ```
   $cert = New-SelfSignedCertificate -CertStoreLocation Cert:\LocalMachine\My -DnsName "IsvCert" -Type CodeSigningCert -KeyExportPolicy Exportable -HashAlgorithm sha256 -KeyLength 2048 -KeySpec Signature -Provider "Microsoft Enhanced RSA and AES Cryptographic Provider" -NotBefore (Get-Date -Year 2020 -Month 1 -Day 1) -NotAfter (Get-Date -Year 2022 -Month 12 -Day 31)
   ```

   b. Get a reference to the new certificate.
c. Create the secure string password that the certificate uses. (Replace "##############" with the certificate password)

```
[String]$certPath = Join-Path -Path "cert:\LocalMachine\My\" -ChildPath "$(($cert.Thumbprint)"
```

```
```

d. Export the certificate private key as .pfx file using the password.

```
Export-PfxCertificate -Cert $certPath -FilePath "C:\Temp\IsvCert.pfx" -Password $certPassword
```

e. Export the certificate public key as a .cer file.

```
Export-Certificate -Cert $certPath -FilePath "C:\Temp\IsvCert.cer"
```

2. Add the certificate to the root store.

```
certutil -addstore root C:\Temp\IsvCert.cer
```
This topic explains how to import independent software vendor (ISV) licenses into an on-premises deployment.

**IMPORTANT**
The process that is described in this topic is available only for customers who have on-premises environments that are deployed with Platform update 12 or later.

For general information about the benefits of ISV licensing, information about how to enable licensing for your solution, and other information that is related to self-signed certificates, see Independent software vendor (ISV) licensing.

**Prerequisites**
Before you import the ISV license file into your on-premises environment, verify that the following prerequisites are met:

- The most recent version of the local agent was used when the environment was deployed.
- The environment is deployed with Platform update 12, and all hotfixes for Platform update 12 are applied. This step is mandatory because Microsoft has released a fix for an ISV licensing scenario. To get the latest set of hotfixes, use the tiles on the **Environment details** page in Microsoft Dynamics Lifecycle Services (LCS).
- Before you import the ISV license, the environment must be deployed, and the Application Object Server (AOS) service must be running.
- Before you import the ISV license, the ISV solution must be applied to the on-premises environment. The ISV solution can be applied to an on-premises environment during the deployment flow. Alternatively, you can use the **Apply updates** flow in LCS to apply the ISV solution as a post-deployment step. If the ISV solution isn't applied before you import the license, the customizations won't be enabled.

**Import licenses**
The following procedure can be used for a sandbox environment or a production environment that is deployed in an on-premises project.

**NOTE**
Because import of an ISV license requires downtime, no business transactions can be performed in the environment during import. When you complete the import, make sure that no one is using the system, and that an official downtime notice has been communicated to all the users.

1. Collect the tenant name and ID for the customer to issue the license to:
   a. Connect to the instance of Service Fabric Explorer where the environment is hosted.
   b. Go to **Clusters** > **Applications** > **AXSFType** > **fabric:\AXSF**, and then, on the right page, select the **Details** tab.
   c. In the **Parameters** table, find the values for the **License_TenantDomainGuid** and
Licence_TenantId keys.

2. Generate a license for the customer (tenant ID and name), and sign the license by using the certificate's private key. The following parameters must be passed to the AXUtil genlicense command to create the license file. The command will generate an XML file.

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>The name of the license file.</td>
</tr>
<tr>
<td>licensecode</td>
<td>The name of the license code from Microsoft Visual Studio.</td>
</tr>
<tr>
<td>certificatepath</td>
<td>The path of the certificate's private key.</td>
</tr>
<tr>
<td>password</td>
<td>The password of the certificate's private key.</td>
</tr>
<tr>
<td>customer</td>
<td>The customer's tenant name.</td>
</tr>
<tr>
<td>serialnumber</td>
<td>The customer's tenant ID.</td>
</tr>
<tr>
<td>expirationdate</td>
<td>Optional: The expiration date of the license.</td>
</tr>
<tr>
<td>usercount</td>
<td>Optional: The number that can be required for the custom validation logic. This number can be the number of users, but it isn't limited to users.</td>
</tr>
</tbody>
</table>

Here is an example of the command.

```
```

3. Copy the licenses that are generated to a folder on one of the machines that is running fabric/AXSF, and verify that fabric/AXSF is healthy.

4. Run the Import-LicensePackage.ps1 script from one of the AOS machines. You can find this script in the latest Deployment scripts folder on the Model tab in the Shared asset library in LCS. Here is a list of the parameters that you must pass to the script:
   - LicenseFilesPath – The path of a folder that contains the license files that must be imported.
   - SqlUser – The same user who is specified in the credentials.json file to run the AOS.
   - SqlPassword – The password that can be used to connect to SQL.
   - EnvironmentConfigPath – The configuration file for the environment. This file is named config.json and is located under the agent share in a folder that has the format wp\<environment-name>\StandaloneSetup.

   After the command is run, log files are generated for each license file that is processed. The names of the log files are in the format (license_file_name).output.log and (license_file_name).error.log. The logs that are generated during database synchronization are located in files that are structured like dbsync.output.log and dbsync.error.log.

5. When the script has been run successfully, validate that the configuration key has been imported and enabled. In the product, the corresponding configuration key will be available and enabled on the License
configuration page. By default, the configuration is enabled. For example, if you added a configuration key that is named ISVConfigurationKey1, it will appear in the list of configuration keys.

When the configuration key is enabled, the changes in the ISV solution will be visible in the product.
Overview of the GDPR

The European Union's General Data Protection Regulation (GDPR) sets a new global standard for privacy rights, security, and compliance for the citizens and residents of the European Union (EU). The GDPR governs the handling and use of personal data of EU citizens and residents. Enforcement of the GDPR begins May 25, 2018, and there are significant consequences for non-compliance. For more information about the regulation, see the European Union site.

NOTE

For information about the scope and coverage of this documentation, see Clarification of the scope of this content section at the end of this topic.

Before utilizing any product features in support of your GDPR compliance efforts, please ensure that you have applied all of the related hotfixes.

The GDPR gives EU citizens specific data subject rights (DSRs) that let them perform the following actions:

- View their personal data.
- Correct errors in their personal data.
- Erase their personal data.
- Object to processing of their personal data.
- Export their personal data.

The GDPR defines personal data in the following way in article 4 of the regulation (organizations do not have personal data):

(1) 'personal data' means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person;

To determine responsibilities for compliance, the GDPR identifies the following roles:

- **Data controller** – The controller controls personal data and determines how it's used. The responsibilities of the controller include but are not limited to collecting, maintaining, directing actions, protecting, modifying and deleting personal data. The controller either adds users to the system, grants access to the system, and collects data from data subjects, or has employees who complete these tasks on the company's behalf. The burden of understanding the process for GDPR requests and carrying out a GDPR request rests with the controller.
- **Data processor** – The processor provides services to, and processes data on behalf of, the data controller. The processor performs actions on behalf of the controller. The processor makes it possible for the controller to be GDPR compliant, but has no ownership of the data and does not respond directly to DSR requests.
- **Data subject** – A data subject is a natural person whose personal information is being used.
- **C1** – C1 is a Microsoft direct customer (IT Admin in the Enterprise Cloud).
C2 – C2 is C1’s customer.

For Finance and Operations apps, Microsoft acts as a processor. As a data processor, Finance and Operations provides processes and features that help you comply with your GDPR obligations as a data controller.

The following illustration shows the flow of data from your customer to the application database, and the roles that you and Microsoft play in that process. For each application, the controller is the tenant administrator, and Microsoft is the processor. In this scenario, the data is sent to the processor (Microsoft), who then processes the data by storing it, retrieving it, sorting it, and so on.

When a data subject chooses to submit a DSR, the data subject makes the request to the controller. Data subjects won’t approach Microsoft to exercise their rights for data that your business has collected. As the processor, Microsoft assists the controller by providing features, or just by making sure that the actions are possible. In other words, the controller accepts and responds to a DSR request, and the processor assists with or enables the compliance request. The following table outlines some of the roles and responsibilities that are relevant.

<table>
<thead>
<tr>
<th>Role</th>
<th>Scenarios</th>
<th>Implementation</th>
<th>Level of data access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Your customer</td>
<td>• View personal data</td>
<td>You must provide a mechanism for your customer to exercise a DSR (process or service).</td>
<td>Your customer sees only their personal data.</td>
</tr>
<tr>
<td>(2)</td>
<td>• Correct personal data</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Erase personal data</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Object to processing</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Export personal data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Your employee – information worker

- View personal data
- Correct personal data
- Erase personal data
- Object to processing
- Export personal data

### Your employee – GDPR administrator

- Validates the user identity request
- Locates the personal data across systems
- Curates the data based on your policy
- Creates a data package or executes an action

- Uses Finance and Operations to locate the data and fulfill the request.
- Writes a customization.
- Reaches out to third parties for shared-controller DSRs.
- Reaches out to Microsoft for activity data.

### You must provide a mechanism for your worker to exercise a DSR (process or service). Some activity information may be obtained from Microsoft

### Your information worker sees only their personal data.

---

**Responding to requests to view, correct, erase, object, or export personal data**

Suppose that a customer decides that they want to understand what personal data of theirs is maintained by an organization. That customer approaches that organization and asks to exercise their DSR. When data subjects exercise their DSRs, controllers must address each of the following items specifically:

- Properly identify the person and role (is the person an employee, a customer, a vendor?) by using information that the data subject gave you as part of their request. This information might be a name, an employee ID or customer number, or another identifier.

- Record the date and time of the request. (You have 30 days to complete the request.)

- Affirm that the DSR request is proper and valid. You will need to work with your legal counsel to determine what is valid. For example, you must make sure that compliance with a DSR request doesn’t conflict with any other legal obligations that you have.

- Verify that you have the information that is related to the request.

**Reasons why certain personal data may not be modified or deleted**

The following table lists several reasons why personal data modification or deletion is restricted in certain situations:
### Right to view

An organization might decide to take any of the following actions in response to a DSR request to view data:

- Use the Person search report to find and collect personal data. To access this report, from the navigation pane, select **Modules > System administration > Inquiries > Person search report**.
- Extend the Person search report by authoring a new entity or extending an existing entity.
- Use search and filter features to find specific personal data and export that data by using the Microsoft Office Export functionality or print that information to a .pdf using browser extensions.
- Use provided documentation to identify data tables that contain data that the controller has identified as personal data.
- Author a custom form that locates and exports personal data.
- Author an external portal or website that allows an authenticated customer to see their personal data.

The Person search report might help you discover personal data that is subject to a DSR request. If the report doesn’t include the information that you’re looking for, check the Microsoft Dynamics Lifecycle Services (LCS) site for possible hotfixes that include the information. You can also extend the report yourself by creating additional entities, or extending the provided entities.

If the Person search report doesn’t contain all the information that the data subject is requesting, you can extend it by using tools that Microsoft has provided. For information about how to extend the Person search report, see [Extend the Person search report](#).

### Right to correct* **

An organization might decide to take any of the following actions in response to a DSR request to correct data:

- Use the Person search report to find and collect personal data.
- Extend the Person search report by authoring a new entity or extending an existing entity.
- Use search and filter features to find specific personal data.
- Author a custom form that locates personal data.
- Author an external portal or website that allows an authenticated customer to correct their personal data.

When data is located, use in-product features to correct the data where the product offers the ability to do so.

*You might find that some data that qualifies as personal data can’t be modified directly. Typically, this data is part of a financial transaction or other business data that is kept “as is” for compliance with financial laws (for example, tax laws), prevention of fraud (such as security audit trail), or compliance with industry certifications.

** GDPR is not a law exclusive of all other laws. As an enterprise resource planning system, Finance and Operations does not allow for modification of certain business or transactional data, and will not endorse nor provide functionality for the modification of business data that is necessary for compliance with other laws or certifications. Finance and Operations will not provide support for modifications/customizations or other actions that result in the corruption of referential or business data integrity.
Right to be forgotten*
An organization might decide to take any of the following actions in response to a DSR request to erase data:

- Delete or otherwise erase personal data where the product enables that action directly.
- Anonymize the personal data where the product enables that action directly.
- Author a customization to erase/modify the personal data.

* GDPR is not a law exclusive of all other laws. As an enterprise resource planning system, Finance and Operations does not allow for deletion of certain business or transactional data, and will not endorse nor provide functionality for the deletion of business data that is necessary for compliance with other laws or certifications. Finance and Operations will not provide support for modifications/customizations or other actions that result in the corruption of referential or business data integrity.

Right to port
An organization might decide to take any of the following actions in response to a DSR request to port data:

- Use the Microsoft Office Add-in to export personal data.
- Author a custom report that enables the export of personal data.
- Author a customization that exports personal data.
- Use or extend the Person search report to gather information in support of a request for a copy of the data subject's personal information.

The Person search report might help you discover personal data that is subject to a DSR request. If the report doesn’t include the information that you’re looking for, check the LCS site for possible hotfixes that include the information. You can also extend the report yourself by creating additional entities.

If the Person search report doesn't contain all the information that the data subject is requesting, you can extend it by using tools that Microsoft has provided. For information about how to extend the Person search report, see Extend the Person search report.

The controller may, at their sole discretion choose to redact certain types of information that may fall outside of the scope of data that must be returned to the data subject as defined within the GDPR.

Right to restrict
An organization might decide to take the following action in response to a DSR request to restrict optional data processing:

- Remove the customer from, for example, a marketing campaign.

* GDPR is not a law exclusive of all other laws. As an enterprise resource planning system, Finance and Operations does not allow for restricted processing of certain business or transactional data, and will not endorse nor provide functionality for the restriction of processing of business data that is necessary for compliance with other laws or certifications. Finance and Operations will not provide support for modifications/customizations or other actions that result in the corruption of referential or business data integrity.

Controller considerations
Controllers can use the following information to complete DSR requests.

System inventory
- Data inventory and tagging – Microsoft has enabled a tagging infrastructure that developers and customers can use. Each data field that is defined in metadata contains a classification property that has a suggested value that the controller can confirm or change to any value or term they choose in order to identify data that they deem fits within the definition of personal data.
Data flow diagram – Microsoft will publish a data flow diagram that identifies flows of data between systems in the customers' production environments.

Person search report – Finance and Operations includes the Person search report, which can be used to gather information in support of a request for a copy of the requestor's personal information.

Activity and diagnostic information

The controller can make DSR requests regarding telemetry data by using the Microsoft Enterprise Privacy Portal. Some telemetry data that we collect is in system generated logs. Without additional information or your assistance, the user's identity is anonymous.

Representation of a person in Finance and Operations

Finance and Operations has a common Global address book. Typically, every time that you add a contact, customer, user, worker, or other person in your system, you first create an address book entry for that person. Each person in the address book is referred to as a party and is assigned a PartyID. The person also takes on a role in the system, such as Customer, User, or Worker, and has a role ID: CustID, UserID, WorkerID, and so on.

![Data flow diagram](image)

Each person is a type of party

Roles that are associated with party records are referred to as party roles. There are several party roles, and they can be assigned to both party types (person and organization):

- **Customer** – An individual, company, or other entity that purchases goods and services that are produced by other individuals, companies, or entities.
- **Prospect** – A party that might be interested in goods or services an organization provides.
- **Worker** – A person who assumes the role of an employee or a contractor, or who is paid in exchange for services.
- **User** – A person who is a user of the system. The user isn't identified in the Global address book.
- **Vendor** – A party that supplies products to one or more legal entities in exchange for payment.
- **Competitor** – A person or organization that provides goods or services that are like the goods or services that your business provides. Out of the box, there is no particular identification for competitors.
- **Applicant** – A person who makes a formal written or electronic request to work for an organization or fill an open position in it.
- **Contact** – A person, either inside or outside your organization, that you've created an entry for. In this entry, you can save information such as the person's street and email addresses, telephone and fax numbers, and webpage URLs.

The right to view and port: It's all about the party
When a data subject approaches the controller to request a copy of their personal data, the controller might choose to use the Global address book information to locate the data that describes the person. As noted in the illustration earlier in this topic, a person is a type of party that plays a role.

Some organizations conduct their activities only through business-to-business relationships and will have modest DSR obligations. By contrast, other organizations conduct their activities through business-to-customer relationships. These organization might choose to use the Global address book and its associative data relationship to write custom reports, custom forms, custom queries, and custom data export features by using the extensibility and customization capabilities and Open in Excel experiences to serve the specific needs of the kinds of data that their business collects from their customers.

The Person search report

To support the controller, this report offers a refinement of the existing entity model reporting functionality that is available in the Data management workspace. The Data management workspace offers a collection of pre-packaged representations of most role types. These representations are known as entities.

NOTE
The Person search report is available for Finance, Supply Chain Management, Commerce, and Human Resources. Currently the report does not support Microsoft Dynamics AX 2012.

An entity represents an instance of a specific role. The data management functionality lets the controller export entity data to several formats, such as colon-separated values, comma-separated values (CSV), semicolon-separated values, tab-separated values, Microsoft Excel, and XML.

The Person search report provides additional capabilities in the Data management workspace that export entity data by providing a party ID that is used to identify all roles (and corresponding entities) that are associated with the party. This capability lets you export all entity and transaction data in a single action, for either a single party or a collection of parties.

When a data subject approaches the controller to request a copy of their personal data, the controller might choose to use the Global address book information to locate the data that describes the person. As noted in the illustration earlier in this topic, a person is a type of party that plays a role.

Some organizations conduct their activities only through business-to-business relationships and will have modest DSR obligations. By contrast, other organizations conduct their activities through business-to-customer relationships. These organization might choose to use the Global address book and its associative data relationship to write custom reports, custom forms, custom queries, and custom data export features by using the extensibility and customization capabilities and Open in Excel experiences to serve the specific needs of the kinds of data that their business collects from their customers.

Additional notes that apply to requests for data

- Data in Management Reporter and in Microsoft Power BI presentations is generated from the information that is entered in various financial documents and then transferred to those applications for reporting purposes. Any request for data should be fulfilled from the financial documents by using tools such as reports, Export to Excel, and the Person search report. You should not need to do additional reporting from Management Reporter or Power BI to fulfill a GDPR request unless you have made customizations that have altered the base functionality.
- Personal data that is included in documents or attachments might also need to be returned to the data subject, independent of any reporting.
- If a master record has transactional data associated with it, it can't be deleted.
- Similarly, transactions that have been posted or completed can't be deleted.
Reasons why Finance and Operations might not support modifying or deleting data out of the box

The following table lists several reasons why data modifications might be restricted.

<table>
<thead>
<tr>
<th>REASON</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit</td>
<td>Data must be preserved for compliance and auditing.</td>
</tr>
<tr>
<td>Calculated</td>
<td>Data that has been calculated can be changed only by changing the data that is included in the calculation.</td>
</tr>
<tr>
<td>Financial, tax, generally accepted accounting principles (GAAP)</td>
<td>Posted transactions can't be modified or deleted.</td>
</tr>
<tr>
<td>Import log</td>
<td>Data must be preserved for compliance and auditing.</td>
</tr>
</tbody>
</table>

What types of personal data might exist in the product

You should expect data requests to come to your company. You can categorize the people who request data into one or, in some cases, more than one relationship with your company:

- Customers
- Vendors
- Workers
- Users
- Warehouse workers
- Truck drivers
- Prospects
- Contacts
- Applicants
- Competitors

Personal data might also be contained in other roles that aren't listed here. Pages used to enter, view or edit personal data have been provided in worksheets for most roles in the preceding list. You can view or download the spreadsheets from the Reference documents for finding and managing personal data page on CustomerSource.

Detailed inventory

As you use Finance and Operations apps, you might find that you generate or collect large amounts of data that resides in multiple data stores. To help you make sense of where your data resides, we've introduced a data marker for each piece of data in our data stores. This marker is called "Asset Classification," and it can be used to identify or track personal data. Any data that you collect has been described as "customer content." Some customer content might contain personal data, and some customer content might contain business data. You can choose to treat all customer content as personal data, or you can change the classification yourself, so that you can identify and track any data that you feel is considered "Personal Data." Although Microsoft has supplied a set of default classifications, you're free to use any classification or identifiers that you choose.
Age Gating: Preventing minors from using the service

Overview
Microsoft mandates that all users of Microsoft software where personal data is collected must use a Microsoft account (MSA) or Microsoft Azure Active Directory (Azure AD) account for authentication. Additionally, those accounts must be configured to enable minors who use the software or service to affirm parental consent for the service to use their personal data.

What is this feature?
As the tenant admin of the service, you will be required to set up Azure AD Age Gating and/or MSA age gating. Any user who isn’t configured by using Azure Age Gating will be restricted from using the service, even if the user isn’t a minor. Age Gating must be configured.

How will age gating work?
The GDPR specifies that systems must stop processing a minor’s personal data if that minor doesn’t have parental consent. Note that consent can be given and then withdrawn. Therefore, a user might have access to the system one day but not the next.

Privacy notices and user subject rights
Displaying your organizations user rights and privacy notice
In the About box, you will find links to the Microsoft user rights documentation, and to the Microsoft privacy and cookies documentation. You can also add a link to your organization’s privacy statement.
On the **System parameters** page, system administrator can add links to the organization's user rights and privacy notices. You can add a valid URL for one or both notice types.

When you've completed your entries in the system parameters, the link to your organization's privacy notice will appear in the **About** box, as shown in the following illustration.
Clarification of the scope of this content

- This documentation is a commentary on the GDPR, as Microsoft interprets it, as of the publication date. We have spent a lot of time with the GDPR and believe that we have been thoughtful about its intent and meaning. But the application of the GDPR is highly fact-specific, and not all aspects and interpretations of the GDPR are well-settled.

- This documentation is provided for informational purposes only, and should not be relied upon as legal advice or to determine how the GDPR might apply to you and your organization. We encourage you to work with a legally qualified professional to discuss the GDPR, how it applies specifically to your organization, and how to best ensure compliance.

- MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY AS TO THE INFORMATION PROVIDED IN THIS PRESENTATION. This documentation is provided “as is.” Information and views expressed in this documentation, including URL and other Internet website references, may change without notice.

- This documentation does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this presentation for your internal, reference purposes only.
This topic provides links to information that can help you respond to a request for information under the General Data Protection Regulation (GDPR) as a customer using Dynamics 365 Finance, Supply Chain Management, Commerce, Human Resources, and Microsoft Dynamics AX 2012.

Your first step in responding to a request for data will usually be to use the Person search report to locate the data that’s requested. In some cases, you might need to use other reports, access specific pages in the product that you’re using, or extend the Person search report. (The report is currently not available for Microsoft Dynamics AX 2012.) This topic points to content that will help you complete those tasks.

Overview

- General Data Protection Regulation overview
- Person search report
- Extend the Person search report
- Manage access to sensitive data

Product-specific considerations

- Respond to requests for personal data in AX 2012
- Respond to requests for personal data in Human Resources
- General Data Protection Regulation overview
- GDPR data requests for Lifecycle Services (LCS)

Compliance Manager

Compliance Manager is a cross-Microsoft Cloud services solution designed to help organizations meet complex compliance obligations like the GDPR. It performs a real-time risk assessment that reflects your compliance posture against data protection regulations when using Microsoft Cloud services, along with recommended actions and step-by-step guidance.

Compliance Manager

You can try Compliance Manager yourself by visiting https://aka.ms/compliancemanager.

Resources

There are a number of resources to help you learn more about Compliance Manager and what it can do you help you meet complex compliance obligations.

- Microsoft 365 - Compliance Manager preview (Office, November 2017)
- Compliance Manager preview is now available (Tech Community, November 2017)
- Get to know the new Service Trust Portal (TechNet, November 2017)
- Announcing Compliance Manager (Tech Community, September 2017)
- Advancing intelligence, management, and security to empower the modern workplace (Office, September 2017)
- New Microsoft 365 features to accelerate GDPR compliance (Microsoft Secure, September 2017)
Additional resources

For more information about the GDPR and the actions that your organization might need to take in response to a request for data, visit the Microsoft Service Trust Portal.
Finance and Operations apps provide a default set of classifications for the kinds of data that are stored in each table. These classifications are subject to change depending on the need to identify different kinds of data. The actual classification for each field in each table can change at any time, depending on differing needs for identifying data.

Through customization, you can change the classification of the following data to meet your own classification and tracking needs.

- **Customer content** – Data collected and managed by the controller (some of which can be personal data).
- **End User Identifiable Information** (EUII) – A natural value used to identify a user of the service.
- **End User Pseudonymous Information** (EUPI) – A generated value used to identify the user of the service.
- **Organizational Identifiable Information** (OII) – A value used to identify the organization using the service.
- **System metadata** – A value that describes the software or used by the software, typically generated by the software.
- **Object metadata** – A value that describes the software or used by the software, but can be provided by the tenant or user of the software.
- **Account data** – A value provided by or used by the tenant to identify the billing information or identify the software used by the tenant.
- **Support data** – Information used to provide customer support.
The Person search report is a refinement of the existing Data management framework of Finance and Operations apps. The Data management framework offers a pre-packaged set of entities that Microsoft authored to identify personal data that is used to define a person and the roles that a person might be assigned to in Finance and Operations applications.

NOTE
You can use the report with Dynamics 365 Finance, Supply Chain Management, Commerce, and Human Resources. The report is not currently available for Microsoft Dynamics AX 2012. The Person search report is available in version 8.0. The report is also available in version 7.3 (delivered via monthly update 7.3.2), in version 7.2 (via KB 4132615), and in version 7.1 (via KB 4132441). The Person search report may be updated periodically. Before using this report, you need to ensure that you have obtained and applied all relevant hotfixes.

You can use the Global address book to create an instance of a person that is described in the data model as a party.

When you add a contact, customer, user, worker, or other person in Finance and Operations data, you typically start by creating an address book entry for that person. Each person in the address book is referred to as a party and is assigned a PartyID. The person also takes on a role in the system, such as customer, user, or worker, and has a role ID: CustID, UserID, WorkerID, and possibly others.

At times, you might want to verify that the information that is entered and used to describe or otherwise identify a person is correct. Situations might also arise where it’s useful to share that information with the data subject who requested the data. The Person search report can help with both these tasks.

The Person search report is extensible. If you find that the existing entities do not contain all of the personal data you are looking for, they can be extended, or new entities can be written. In addition, you can change the data mappings for each entity and remove fields that you don’t want to export.

The Person search report lets you specify different identifiers for a person, such as a CustomerID or VendorID. It
will then collect, filter, and populate the entity collection set with personal data that is related only to the person you specified.

On rare occasions, a single person might be entered in your system more than once. The Person search report lets you specify each person instance to be included on a single report. For example, someone named Fred Smith might be both “Fred Smith” and “F. D. Smith” in your address book.

An individual might exist as multiple parties in data. You can provide multiple identifiers for each party type, and each party type’s personal data will be included on a single report.

**Download the default template**

The Person template contains a list of the entities that will be used to download information. The template must be loaded before the Person search report can be used. The template can be loaded from within the Templates form in Data management for versions 7.2 and later. To download templates from Data management, complete the following steps.

1. Open the Data management workspace.
2. If this is the first time that the workspace has been opened, it will load all of the data entities. You must load all the data entities before you load the template.
3. Click the Templates tile.
4. Select the Load default templates button.
5. Select Person search.
6. Click Load selected.

You can also download a template from LCS and import it for versions 7.1 or later. To do so, complete the following steps.

1. Log in to LCS.
2. Click the Shared asset library tile.
3. Select the Data package asset type.
4. Click the template named Template-x.x-Person search, where xx is the application version that you’re using, and download it.
5. Open the Data Management workspace.
6. If this is the first time that the workspace has been opened, the workspace will load all of the data entities. All entities loaded before you download the template.
7. Click on the Templates tile.
8. Create a new template called Person search.
9. Click Import template.
10. Browse to the template and click Upload.
11. Click OK to import the template.

**Generate a person search**

To use the Person search report, you must complete these tasks.

1. From the System administration menu, open the Person search list page, and create a new search.
2. The search gives you three options: you can search by ID, by name, or by address. Add the type of search that you want.

3. Run the search to show the results.

4. Verify that the results are valid. Clear any selections that return information that you don't want to include on the report.
5. Select Process report, and then select the Person search template.

6. Select OK. A data package is generated.

7. When the package has been generated, export it to your selected data format.

**NOTE**

Documents that are attached to records are not included in the data export. Attachments must be manually downloaded and shared with the individual who requested personal data.

**Additional resources**

You can learn more about the GDPR on the [European Union's website](https://www.europa.eu), from information on the [Microsoft Trust Center](https://www.microsoft.com) and in the [General Data Protection Regulation overview](https://www.microsoft.com).
The Person search report for Finance and Operations apps is backed by an intelligent search processor that is designed to manage a collection of entities for a single person. The Person search report searches Finance and Operations data and creates a set of resulting identifiers. Each result references a search category (for example, Customer) and a result record in a related table. For information about using the Person search report, refer the Person search report topic.

**NOTE**
The Person search is available for Dynamics 365 Finance, Supply Chain Management, Commerce, and Human Resources. The Person search report is not currently available for Microsoft Dynamics AX 2012.

**Add another entity to the default template**
You can add any entity to the default Person search report template. Open the Data management workspace, and select Templates. Add the entity to the template. The entity that you add must include at least one of the fields that are used to filter the Person search report.

**Create a new search category**

1. Use the PersonSearchResultCategory enumeration to distinguish different categories of results, such as workers versus applicants.
2. Extend the PersonSearchResultCategory enumeration as needed to create new result types.

**Create search processing for the new search category**

In this example, you will create a new processor class.

1. Extend the PersonSearchModule enumeration with a new search area.
2. Create a class that extends the PersonSearchProcessor class and includes the PersonSearchProcessorFactoryAttribute attribute, with the new person search module area as a parameter.
3. In the PersonSearchProcessor extended class, override the doSearch method with your desired search logic. As shown in the following example, extend the PersonSearchResult table to create new table relationships.

```csharp
PersonSearchResultCategory::Customer needs a relation:PersonSearchResult.ResultRecId = CustTable.RecId, PersonSearchResult.ResultTableId = CustTable.TableId
```

**Integrate with the Global address book (Optional)**
If you want to integrate with the Global address book, insert any discovered party numbers into the PersonSearchPartyNumberTmp table. The findPartyLink method of PersonSearchProcessor tries to link party numbers to other search artifacts, such as users, customers, or vendors.

This method has a delegate, onFindPartyLink, that lets you specify additional artifacts to search, based on the
party numbers.

The PersonSearchCriteria tables let end users specify the parameters by which to search the system for personal data.

Create new search criteria

If you need new search criteria, follow these steps.

1. Extend the PersonSearchCriteriaName, PersonSearchCriteriaAddress, and PersonSearchCriteriaKnownId tables, or create your own tables.
2. Extend the PersonSearchDialog form to show these new data fields.
3. Use the new criteria during search processing.

The PersonSearch form shows the set of results that was discovered by a person search.

Show the new search category in the PersonSearch form

1. Create a new view on the PersonSearchResult table. This view should restrict results to only your new PersonSearchResultCategory.
2. Join the view to your result record, and create the view fields that are needed (for example, PersonSearchResultCustomerView).
3. Extend the PersonSearchResult table to create a new relationship to the new view. This relationship should join on the person search ID.
4. Extend the PersonSearch form:
   a. Add the new view as a data source.
   b. Add a new tab to the results with the result grid and the Include/Exclude buttons.
   c. Create event handlers for the Include/Exclude buttons to update the PersonSearchResult records.

   The PersonSearchEventHandler::updateMarkedOnButtonClicked() method is provided for convenience.

NOTE
If you want to see the record count in the result caption, create an event handler on the OnQueryExecuted view data source event. Next, call the setResultCountOnGridCaption() method on the PersonSearch form to update the count.

Each PersonSearchEntityFilterRelation record specifies the conditions when a filter should apply, and the filter table and field to apply.

The set of filter relations is compared to the template metadata when the package is built.

For a filter to be created, a PersonSearchResult record with the matching filter category must exist. After it's found, the PersonSearchResult references the table field where the filter value resides.

Create new filters

1. Use the Chain of Command to extend the PersonSearchEntityFilterRelation table.
2. Decide on the type and source of the new filter:
a. If the filter is an extended data type (EDT) or enumeration, set the MetadataTypeId to the data type ID.
b. If the filter is a source table field, specify the source table and source field IDs.
c. If the filter is an entity field, specify the entity field ID.
3. Decide on the filter table and filter field.
   The filter table must be available as a PersonSearchResult with a matching category. Otherwise, no filters will be created.
4. Insert the new filter record.
   The person search framework will automatically manage initialization of all exclusions when the form is first opened. Exclusions let you suppress the filter building functionality for specific entity fields.

Create new exclusions

1. Use the Chain of Command (COC) to extend the PersonSearchEntityExclusion table.
2. In the CoC method, specify the entity, the entity field, and whether the exclusion is active.
3. Insert the new exclusion record.
   The person search framework will automatically manage initialization of all exclusions when the form is first opened.

Additional resources

If you're extending the Person search report as part of a response to a request for data under the General Data Protection Regulation (GDPR) in the European Union, more information about that regulation is available in the General Data Protection Regulation overview.

You can learn more about the GDPR on the European Union's website and on the Microsoft Trust Center.

Disclaimer

(c)2018 Microsoft Corporation. All rights reserved. This document is provided “as-is.” Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it. This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.
Manage access to sensitive data

System administrators can use the User log page to keep an audit log of users who have logged on to the system. Knowing who has logged in can help protect your organization's data. We've enhanced the user logging capability to let the administrator identify roles that provide access to sensitive data.

What is sensitive data?
An organization can define what constitutes sensitive data in whatever way serves its needs. For some organizations, sensitive data might be any data that is related to financial or human resource data, or just data that is personal data. Some industries or some countries or regions might have a more specific definition of sensitive data that an organization can adopt for itself. It's up to each organization to decide whether and how to use the sensitive data identifier.

The sensitive data identifier enhances the user logging experience by letting your organization produce audit logs that show who in your system has access to sensitive data. This capability is helpful for organizations that might have multiple roles that have varying degrees of access to certain data. It can also be helpful for organizations that want a detailed level of auditing to track users who have had access to data that's been identified as sensitive data.

Language-specific information
The role information in the user log is language-specific and matches the current user language.
Log retention

The log entries of users who have access to data that’s been declared to be sensitive data can be retained separately from all other data in the log. The administrator can enable this functionality by setting an option on the User log cleanup page.

User log cleanup

Parameters

<table>
<thead>
<tr>
<th>History limit (days)</th>
<th>Keep log entries with role information</th>
<th>Records to include</th>
<th>Run in the background</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTE

This feature is available in version 8.0. This feature is available for Dynamics AX 2012 R3 (via KB 4074643)

Additional resources

You can learn more about the GDPR on the European Union's website and on the Microsoft Trust Center.

Disclaimer

(c)2018 Microsoft Corporation. All rights reserved. This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it. This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.
This topic can help both businesses that use Microsoft Dynamics 365 Human Resources, and also partners and independent software vendors (ISVs), when they comply with data subject rights (DSR) requests. For more information about the European Union's General Data Protection Regulation (GDPR) and the related resources that Microsoft provides, see General Data Protection Regulation overview.

For Human Resources, Microsoft acts as a processor. As a data processor, Human Resources provides processes and features that let you comply with your GDPR obligations as a data controller.

Rights

Data subjects have the following rights under the GDPR, and a data controller might take any of the actions that are listed under each right in response to a DSR request.

Right to view

- Use the Person search report to find and collect personal data that is subject to a DSR request. For information about using this report, see the Person search report topic.
- Use advanced search and filters to find specific personal data and export that data by using the Microsoft Office Export functionality.
- Extend the Person search report by adding an existing entity. For information that can help you extend the report, see Extend the Person search report.

Right to modify*

- Use advanced search and filters to find the data that should be corrected, and correct the data directly in Human Resources.

* You might find that some data that qualifies as personal data can't be modified directly in the product or feature. Typically, this data is part of a financial transaction or other business data that is kept "as is" for compliance with financial laws (for example, tax laws), prevention of fraud (such as security audit trail), or compliance with industry certifications. As the controller, it's your responsibility to correct inaccurate or incomplete personal data.

Right to be forgotten*

- You can delete or erase personal data where the product enables that action directly. As the controller, you should ensure any personal data that a data subject requests be erased does not conflict with other compliance obligations your organization may have around data retention, for example proof of payment or proof of tax.

* You might find that some data that qualifies as personal data can't be modified directly in the product or feature. Typically, this data is part of a financial transaction or other business data that is kept "as is" for compliance with financial laws (for example, tax laws), prevention of fraud (such as security audit trail), or compliance with industry certifications. As the controller, it's your responsibility to correct inaccurate or incomplete personal data.

Right to port

The following options are available to help you port personal data in response to a data rights request.
- Use the Microsoft Office Add-in to export personal data.
- Author a custom report that enables the export of personal data.
- Use or extend the Person search report to gather information in support of a request for a copy of the data subject's personal data.

**Right to restrict processing**
- Remove the employee from, for example, a course.
- Following guidance from an organization’s legal counsel, the company might refuse the right to restrict processing where data is needed by the company for compliance with other legal or industry mandates.

**Additional notes that apply to requests for personal data**
- Personal data that is found in Microsoft Power BI is generated from the information that is entered in Human Resources and then transferred to that application for reporting purposes. Any request for personal data should be fulfilled from the information in Human Resources, by using tools such as reports, Export to Excel, and the Person search report. You should not need to do additional reporting from Power BI to fulfill a DSR request.
- Human Resources doesn’t export documents that are attached to records. These attachments must be manually downloaded and shared with the individual who has made the DSR.
- If transactional data is associated with a master record, that record can’t be deleted.
- Similarly, transactions that have been posted or completed can’t be deleted.

**Reasons why certain personal data may not be modified or deleted in Human Resources**
The following table lists several reasons why personal data modification or deletion is restricted in certain scenarios.

<table>
<thead>
<tr>
<th>REASON</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Financial, tax, generally accepted accounting principles (GAAP)</td>
<td>A party can't be deleted, but the party's name can be updated.</td>
</tr>
<tr>
<td>Financial, tax, GAAP</td>
<td>A current worker's data can't be deleted, but the worker's name can be updated.</td>
</tr>
<tr>
<td>GAAP</td>
<td>Posted or completed transactions can't be modified.</td>
</tr>
</tbody>
</table>

**Additional information**
Only terminated workers can be deleted from Human Resources. Follow these steps to delete terminated workers.

- Delete position assignments.

  To delete a position assignment, select **Position assignments** on the **Worker** page. Select **As of date** and select **Display all records**. Drill into the position number, select **Changes timeline > Manage changes > Position worker assignments**, and remove the position assignment record that is associated with the worker that you’re deleting.

- Delete fixed compensation.

  To delete fixed compensation, select **Employment history** on the **Worker** page. Select **Employment history > Employment > Fixed compensation**, and delete the fixed compensation plans for the worker.
Delete variable compensation enrollments.

To delete variable compensation, select **Employment history** on the **Worker** page. Select **Employment history > Employment > Variable compensation plan enrollment** and delete the variable compensation plan enrollments for the worker.

Delete any associated checklists.

To delete the checklists, select the **Checklists** option on the **Worker** page.

Compensation isn't assigned to contractors. Therefore, those steps can be skipped in the preceding process.


**Additional resources**

You can learn more about the GDPR on the [European Union’s website](https://www.europa.eu/) and on the [Microsoft Trust Center](https://www.microsoft.com/trustcenter).

**Disclaimer**

(c)2019 Microsoft Corporation. All rights reserved. This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it. This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.
This topic can help businesses that use Microsoft Dynamics AX 2012, implementation partners, and independent software vendors (ISVs) comply with requests from a data subject concerning their personal data under the European Union’s General Data Protection Regulation (GDPR). For more information about the GDPR and the related resources that Microsoft provides, see the General Data Protection Regulation overview. Although the “Guide to the GDPR for Finance and Operations” was written for Finance and Operations apps, the rights that a data subject has under the GDPR apply to organizations using Dynamics AX 2012. The steps listed in the “Guide to the GDPR for Finance and Operations” can be used by an organization responding to a request for personal data and are also appropriate for organizations using Dynamics AX 2012, with the exception of the Person search report, which is not available for Dynamics AX 2012. This topic lists additional points that are specific to Dynamics AX 2012.

Applicable product updates for Dynamics AX 2012

Additional product updates and information related to GDPR that’s specific to AX 2012 are available from Microsoft Dynamics Lifecycle Services (LCS). Sign-in is required for access to LCS.

Here are some example inquiries and reports that are available in AX 2012 that might help you find and report personal data:

- Home > Common > Global address book
  
  In the inquiry window, enter a person's name in the search box.

- Accounts payable > Common > Vendor > All vendors

- Accounts receivable > Common > Customer > All Customers

- Human resources > Common > Workers

- Sales and marketing > Common > Customers, Prospects, Leads, or Contacts

Additional resources

You can learn more about the GDPR on the European Union's website and on the Microsoft Trust Center.

Disclaimer

(c)2018 Microsoft Corporation. All rights reserved. This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it. This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.
Overview

This topic offers information for complying with requests for data under the General Data Protection Regulation (GDPR) if the request requires accessing data in Microsoft Lifecycle services (LCS). For general information about this regulation and the resources Microsoft is providing to support compliance with it, see General Data Protection Regulation overview.

User management

A tenant administrator can view and manage users on the Organization users tile on the home page in LCS.

An administrator can view and manage all organization users on the Organization user page in LCS.

To export a user, the administrator can use filtering to find a specific user, select that user, and copy the name and email address of the user.

Additional resources

- Official GDPR site
- Microsoft Service Trust Portal
Lifecycle Services (LCS) for Microsoft Dynamics is a collaboration portal that provides an environment and a set of regularly updated services that can help you manage the application lifecycle of your implementations of the Dynamics 365 Finance and Operations apps.

- What's new in Lifecycle Services (LCS)
- Lifecycle Services (LCS) user guide
- Projects in Lifecycle Services (LCS)
- Project onboarding
- Methodologies in Lifecycle Services (LCS)
- Business process modeler (BPM) in Lifecycle Services (LCS)
- Cloud-hosted environments in Lifecycle Services (LCS)
- Manage the support experiences for Finance and Operations apps
- Configuration in Lifecycle Services overview
- Customization analysis in Lifecycle Services (LCS)
- Infrastructure estimator in Lifecycle Services (LCS)
- Issue search in Lifecycle Services (LCS)
- License sizing estimator in Lifecycle Services (LCS)
- Request for proposals (RFP) responses
- System diagnostics in Lifecycle Services (LCS)
- Upgrade analysis in Lifecycle Services (LCS)
- Usage profiler in Lifecycle Services (LCS)
- Downloadable tools in Lifecycle Services (LCS)

**Additional resources**

- For information about how to contact Microsoft if you have technical questions about Dynamics 365 Finance and Operations apps, or if you need help accessing Microsoft Dynamics Lifecycle Services (LCS), see [Get support for Finance and Operations apps](#) or Lifecycle Services (LCS).

- For information about how to contact Microsoft if you have technical questions about Microsoft Dynamics AX 2012 or need support, see [Manage the support experiences for Finance and Operations apps](#).
LCS workspace for the current versions of the Finance and Operations apps

When you sign up for the current versions of Finance and Operations apps, your subscription includes an Implementation project workspace. After you activate the service, the tenant administrator must sign in at https://lcs.dynamics.com by using the tenant account. The project workspace is automatically created for your organization. The workspace includes the following elements:

- Enabled features, based on the offer that you selected
- Environments that are deployed and managed by Microsoft
- Guidance that is provided through the Action center to help you complete required actions
- A new methodology experience that includes tasks that lock as you move through the implementation
- A more complete history that specifies who completed each methodology phase and task
- Milestones that you can use to track critical project dates
- Various services to help you with your implementation

Methodologies

As a customer, you must complete the steps that are outlined in the methodology to gain access to the production environment. Before a phase can be marked as completed, you must complete the specified mandatory tasks. Locked tasks, such as tasks 1.6 and 1.9 in the following screenshot, are unlocked after you've completed the required actions. To learn which actions must be completed before a specific task can be unlocked, click the lock icon for that task.
In the case of prerequisites, after you complete the required tasks, you can mark the dependent tasks as completed. For example, in the following screenshot, tasks 1.6 and 1.9 depend on task 1.5. Because task 1.5 has now been completed, the two dependent tasks can be marked as completed.

**Milestones**

High-level milestones must be defined for a project. Milestones can help you track the deliverables that must be completed and your progress toward the milestone goals. Color indicators help you quickly learn whether you’re behind schedule. For example, in the following screenshot, the milestones are yellow. To enter or update the milestone dates, click the diamond shape in the methodology, and then click the **Edit** button (pencil icon). You can change milestone dates at any time.
When you’ve finished entering milestones, the Publish plan and milestone task opens, and you can mark it as completed.

Set up milestones

Select the date on which each milestone must be completed. These dates will enable and disable capabilities in Lifecycle Services.

<table>
<thead>
<tr>
<th>Milestone</th>
<th>End date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>1/6/2016</td>
</tr>
<tr>
<td>Design</td>
<td>2/9/2016</td>
</tr>
<tr>
<td>Test</td>
<td>3/17/2016</td>
</tr>
</tbody>
</table>
When you've completed all the required tasks in a phase, you can click **Complete phase** to mark the phase as completed. After you mark a phase as completed, next steps become available in Microsoft Dynamics Lifecycle Services (LCS).

**Methodology description and history**

Descriptions can help you understand what is expected of you for a specific methodology task or phase. You can expand the methodology description to learn more about each task, and then collapse the description when you've finished. The task and phase history can tell you when a task or phase was completed or reopened. If you're a project manager, this information can help you stay on top of the high-level tasks that are required for your implementations.
You can use the Subscription estimator tool to evaluate your subscription requirements for the current versions of the Finance and Operations apps. To use Subscription estimator, download the usage profile, which is a Microsoft Excel workbook. Then, in the workbook, complete the following worksheets:

- Deployment details
- Instance Characteristics
- Retail & Commerce

After you’ve completed the worksheets, enter the data from the summary sheet into Subscription estimator by clicking **New estimate**. You must make one estimate the active estimate. Make sure that the estimate that you mark as active is same as the offer that you bought through the VL or CSP channel.

### New deployment experience

To provision your environment, you must to complete a configuration checklist. As you make progress through the methodology, environments become available to you. Click **Configure** to add deployment information.

Because the information that you enter determines your experience, carefully review your input. After you’ve entered all the required information, sign-off is required for the deployment request. The user who completes the sign-off becomes the system administrator on the instance. Verify that the correct user completes the sign-off for the deployment. After the sign-off is completed, the Microsoft site reliability engagement team reviews the request. After the team has reviewed the information that you entered, it initiates the provisioning. If the information isn’t correct, the team will contact you. After the provisioning is completed, the status is updated to indicate that the environment has been deployed, as shown in the following screenshot. If the provisioning takes longer than expected, the Microsoft site reliability engagement team reviews the status and takes appropriate actions. These actions might include contacting you. After the environment is provisioned, click **Full details** to open the Detailed environment page, where you can sign in to the system, view the monitoring status, or
view relevant updates.

<table>
<thead>
<tr>
<th>SANDBOX: DEVELOP AND TEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Checkmark] Environment DynDemoDevTest is deployed</td>
</tr>
</tbody>
</table>

[Full details]
This article explains how partners can get started with Microsoft Dynamics Lifecycle Services (LCS).

As a Finance and Operations partner, you can access the current version by following the steps in Sign up for preview subscriptions.

Projects for partners in LCS
After you sign up as a partner for the current version, you can create two types of projects:

- Prospective presales
- Migrate, create solutions, and learn the Dynamics 365 Finance and Operations apps

Prospective presales project
Work with new prospects to help them understand the business processes that are available, and to help them evaluate their subscription needs. Note that only partners can provision a new cloud environment.

Migrate, create solutions, and learn the project for Finance and Operations apps
Create a Microsoft Dynamics AX 2012 project workspace. You can use the project to support the upgrade from Dynamics AX 2012 to the current version. You can also use the project to learn how to use the current version or create solutions in Microsoft Dynamics AX Lifecycle Services (LCS).

Projects for customers in LCS
For every customer who signs up for LCS, an Implementation project workspace is automatically created during the sign-up process. As a partner, you can’t create an Implementation project. For more information about the Implementation project workspace, see Lifecycle Services (LCS) for Finance and Operations apps customers.

Services within project workspaces behave in the same manner. However, an important difference between an Implementation project and other project types is the ability to configure the current version of Finance and Operations, which is managed by Microsoft.
You can start and stop environments through Microsoft Dynamics Lifecycle Services (LCS) via the LCS Environment API. Using these APIs will ensure the LCS environment status is synced with the actual environment.

Note that the same validation rules from the details page in LCS apply to the API.

**NOTE**
- Only **Customer-managed** environments are supported. Self-service environments do not have the same concept of stop and start and are not supported by this API. Microsoft-managed environments are not supported.
- These APIs will trigger/invoke the operation. A successful response only indicates that the trigger was successful.
- For **stop**, non-success will be returned if the environment is already undergoing another operation or if the environment is already stopped.
- For **start**, non-success will be returned if the environment is already undergoing another operation but will return success if the environment is already started.

### Permissions

One of the following permissions is required to call this API. For more information about permissions and how to select them, see the Database Movement API Authentication content.

<table>
<thead>
<tr>
<th>PERMISSION TYPE</th>
<th>PERMISSIONS (FROM LEAST PRIVILEGED TO MOST PRIVILEGED)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delegated (work or school account)</td>
<td>user_impersonation</td>
</tr>
</tbody>
</table>

### HTTP request

Use the following POST method to send an HTTP request to stop or start an environment.

**Stop an environment**

```
POST /environment/v1/stop/project/{projectId}/environment/{environmentId}
```

**Start an environment**

```
POST /environment/v1/start/project/{projectId}/environment/{environmentId}
```

### Request headers

Use the following header value in the HTTP request header.

<table>
<thead>
<tr>
<th>HEADER</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization</td>
<td>Bearer (token) (required)</td>
</tr>
</tbody>
</table>
Request body
Don't supply a request body for this method.

Response
The response is always a **200 OK** response, unless you aren't correctly authenticated. Be sure to use the **IsSuccess** property to evaluate the success or failure of the action.

Example
Request to stop an environment

```
POST /environment/v1/stop/project/{projectId}/environment/{environmentId}
```

Successful response

```
{
   "IsSuccess": true,
   "OperationActivityId": "55eb4327-9346-4c7b-82bd-fe8ef15112c6",
   "ErrorMessage": null,
   "VersionEOL": "9999-12-31T23:59:59.9999999"
}
```

Rate limits
To better load balance the request, there are rate limits on the Start and Stop API:

For **Start** API, the following limits will be enforced:

- 1 call for each environment for 5 minutes
- 30 calls for each user for 30 minutes

For **Stop** API, the following limits will be enforced:

- 1 call for each environment for 5 minutes
- 30 calls for each user for 30 minutes

**NOTE**
Requests that exceed the limits will be rejected with a "HTTP 429 Too Many Requests" response. The **retry-after** header will indicate the number of seconds when the request can be retried.
You can fetch environment metadata through Microsoft Dynamics Lifecycle Services (LCS) via the LCS Environment API. This API returns a paginated list that, by default, includes all environments in the project. The optional query string parameters can be used to filter the response.

Permissions

**API application**

One of the following permissions is required to call this API. For more information about permissions and how to select them, see Database movement API - Authentication.

<table>
<thead>
<tr>
<th>PERMISSION TYPE</th>
<th>PERMISSIONS (FROM LEAST PRIVILEGED TO MOST PRIVILEGED)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delegated (work or school account)</td>
<td>user_impersonation</td>
</tr>
</tbody>
</table>

**LCS**

In LCS, the user who is used in the API OAuth authentication must be added to the project as either a project owner or an environment administrator. The user must accept the invitation to the project.

**HTTP request**

Use the following GET endpoint to fetch environment metadata.

**Fetch metadata for all environments in a project**

```plaintext
GET /environmentinfo/v1/detail/project/{projectId}/?page=1
```

**Fetch metadata for a single environment by ID**

```plaintext
GET /environmentinfo/v1/detail/project/{projectId}/?environmentId={environmentId}
```

**Fetch metadata for a single environment by name**

```plaintext
GET /environmentinfo/v1/detail/project/{projectId}/?environmentName={environmentName}
```

**Request headers**

Use the following header values in the HTTP request header.

<table>
<thead>
<tr>
<th>HEADER</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization</td>
<td>Bearer (token) (required)</td>
</tr>
<tr>
<td>'x-ms-version'</td>
<td>'2017-09-15' (required)</td>
</tr>
</tbody>
</table>
Request body
Don't supply a request body for this method.

Response

HTTP
The response is always a "200 OK" response, unless you aren't correctly authenticated. Be sure to use the IsSuccess property to evaluate the success or failure of the action.

Pagination
The result includes a Boolean ResultHasMorePages property that indicates whether another page of results is available. The ?page= query string parameter can be used to fetch a specific page.

Data
For each environment, the following properties are available. If no value is available for a property, null is returned.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnvironmentId</td>
<td>The LCS environment ID.</td>
</tr>
<tr>
<td>EnvironmentName</td>
<td>The environment name.</td>
</tr>
<tr>
<td>ProjectId</td>
<td>The ID of the LCS project that contains the environment.</td>
</tr>
<tr>
<td>EnvironmentInfrastructure</td>
<td>The infrastructure type of the environment (for example, SelfService or MicrosoftManaged).</td>
</tr>
<tr>
<td>EnvironmentType</td>
<td>The environment type (for example, Production or Sandbox).</td>
</tr>
<tr>
<td>EnvironmentGroup</td>
<td>The environment group (for example, Primary or DisasterRecovery).</td>
</tr>
<tr>
<td>EnvironmentProduct</td>
<td>The product that is running in the environment.</td>
</tr>
<tr>
<td>EnvironmentEndpointBaseUrl</td>
<td>The base URL of the environment.</td>
</tr>
<tr>
<td>DeploymentState</td>
<td>The state of the most recent environment operation.</td>
</tr>
<tr>
<td>TopologyDisplayName</td>
<td>The product topology that is deployed in the environment.</td>
</tr>
<tr>
<td>CurrentApplicationBuildVersion</td>
<td>A string of the application version.</td>
</tr>
<tr>
<td>CurrentApplicationReleaseName</td>
<td>A string of the application release name.</td>
</tr>
<tr>
<td>CurrentPlatformReleaseName</td>
<td>A string of the platform version.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CurrentPlatformVersion</td>
<td>A string of the platform release name.</td>
</tr>
<tr>
<td>DeployedOnUTC</td>
<td>A Coordinated Universal Time (UTC) date/time value that indicates when the environment was deployed.</td>
</tr>
<tr>
<td>CloudStorageLocation</td>
<td>The primary Azure location of the environment.</td>
</tr>
<tr>
<td>DisasterRecoveryLocation</td>
<td>The secondary Azure location of the environment.</td>
</tr>
<tr>
<td>DeploymentStatusDisplay</td>
<td>The current status of the environment.</td>
</tr>
<tr>
<td>CanStart</td>
<td>A Boolean value that indicates whether the environment can be started.</td>
</tr>
<tr>
<td>CanStop</td>
<td>A Boolean value that indicates whether the environment can be stopped.</td>
</tr>
</tbody>
</table>

**Example response**

Successful response for a project-level request
To better load balance requests, there are rate limits on this API:

- 6 calls for each project per minute
Requests that exceed the limits will be rejected, and an "HTTP 429 Too Many Requests" response will be returned. The `retry-after` header will indicate the number of seconds that the request can be retried after.
You can fetch environment history metadata through Microsoft Dynamics Lifecycle Services (LCS) via the LCS Environment API. This API returns a paginated list that includes ongoing and past operations.

## Permissions

### API application

One of the following permissions is required to call this API. For more information about permissions and how to select them, see [Database movement API - Authentication](#).

<table>
<thead>
<tr>
<th>PERMISSION TYPE</th>
<th>PERMISSIONS (FROM LEAST PRIVILEGED TO MOST PRIVILEGED)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delegated (work or school account)</td>
<td>user_impersonation</td>
</tr>
</tbody>
</table>

### LCS

In LCS, the user who is used in the API OAuth authentication must be added to the project as either a project owner or an environment administrator. The user must accept the invitation to the project.

## HTTP request

Use the following GET endpoint to fetch environment history for a given environment.

```plaintext
GET /environmentinfo/v1/history/project/{projectId}/environment/{environmentId}/?page=1
```

## Request headers

Use the following header values in the HTTP request header.

<table>
<thead>
<tr>
<th>HEADER</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization</td>
<td>Bearer (token) (required)</td>
</tr>
<tr>
<td>'x-ms-version'</td>
<td>'2017-09-15' (required)</td>
</tr>
<tr>
<td>Content-Type</td>
<td>application/json</td>
</tr>
</tbody>
</table>

## Request body

Don't supply a request body for this method.

## Response

### HTTP

The response is always a "200 OK" response, unless you aren't correctly authenticated. Be sure to use the IsSuccess property to evaluate the success or failure of the action.
Pagination
The result includes a Boolean ResultHasMorePages property that indicates whether another page of results is available. The ?page= query string parameter can be used to fetch a specific page.

Data
For each history operation, the following properties are available. If no value is available for a property, null is returned.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The supplied operation history name.</td>
</tr>
<tr>
<td>Type</td>
<td>The operation type.</td>
</tr>
<tr>
<td>TypeDisplay</td>
<td>The display string for the operation type.</td>
</tr>
<tr>
<td>StartDateTimeUtc</td>
<td>The start date and time of the operation in Coordinated Universal Time (UTC).</td>
</tr>
<tr>
<td>EndDateTimeUtc</td>
<td>The end date and time of the operation in UTC.</td>
</tr>
<tr>
<td>Status</td>
<td>The status of the operation.</td>
</tr>
<tr>
<td>ActivityId</td>
<td>The globally unique identifier (GUID) for the operation's activity.</td>
</tr>
<tr>
<td>EnvironmentId</td>
<td>The ID of the environment that the operation was performed against.</td>
</tr>
<tr>
<td>ProjectId</td>
<td>The ID of the project that the operation was performed against.</td>
</tr>
</tbody>
</table>

Example response
Successful response
Rate limits

To better load balance the requests, there are rate limits on this API:

- 6 calls for each environment every 30 seconds
- 6 calls for each project per minute

**NOTE**

Requests that exceed the limits will be rejected, and an “HTTP 429 Too Many Requests” response will be returned. The `retry-after` header will indicate the number of seconds that the request can be retried after.
You can fetch the Regression Suite Automation Tool (RSAT) certificate bundle for an environment through Microsoft Dynamics Lifecycle Services (LCS) via the LCS Environment API. This API returns a Base 64–encoded zip file and a Base 64–encoded password for the private certificate password.

The full process for consuming the zip can be found on the Regression Suite Automation Tool installation and configuration page.

Permissions

API application

One of the following permissions is required to call this API. For more information about permissions and how to select them, see Database movement API - Authentication.

<table>
<thead>
<tr>
<th>PERMISSION TYPE</th>
<th>PERMISSIONS (FROM LEAST PRIVILEGED TO MOST PRIVILEGED)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delegated (work or school account)</td>
<td>user_impersonation</td>
</tr>
</tbody>
</table>

LCS

In LCS, the user who is used in the API OAuth authentication must be added to the project as either a project owner or an environment administrator. The user must accept the invitation to the project.

HTTP request

Use the following GET endpoint to fetch the zip file for an environment's RSAT certificate.

Fetch the RSAT certificate by environment

GET `/environmentinfo/v1/rsatdownload/project/{projectId}/environment/{environmentId}`

Request headers

Use the following header values in the HTTP request header.

<table>
<thead>
<tr>
<th>HEADER</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization</td>
<td>Bearer (token) (required)</td>
</tr>
<tr>
<td>'x-ms-version'</td>
<td>'2017-09-15' (required)</td>
</tr>
<tr>
<td>Content-Type</td>
<td>application/json</td>
</tr>
</tbody>
</table>

Request body

Don't supply a request body for this method.
Response

HTTP
The response is always a "200 OK" response, unless you aren't correctly authenticated. Be sure to use the IsSuccess property to evaluate the success or failure of the action.

Data

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CertificateZipEncoded</td>
<td>A zip containing the .PFX and .CER files in a Base 64-encoded byte array.</td>
</tr>
<tr>
<td>CertificateSecretEncoded</td>
<td>The private certificate's private secret as a Base 64-encoded string. This will change every request.</td>
</tr>
<tr>
<td>ExpirationDateTimeUTC</td>
<td>A date and time in UTC of when the certificate is not valid after.</td>
</tr>
<tr>
<td>Filename</td>
<td>The filename of the zip being returned.</td>
</tr>
</tbody>
</table>

Example response

Successful response for a project-level request

```
{
    "Data": {
        "CertificateZipEncoded": "<base 64-encoded zip>",
        "CertificateSecretEncoded": "<base 64-encoded password>",
        "ExpirationDateTimeUTC": "Thursday, June 30, 2022 8:52:13 PM",
        "Filename": "RSATCertificate_TestEnv1_20210805-100102.zip"
    },
    "IsSuccess": true,
    "OperationActivityId": "2234bff0-432d-478b-a5ac-1ccb529ee608",
    "ErrorMessage": null,
    "VersionEOL": "9999-12-31T23:59:59.9999999"
}
```

Parsing data via PowerShell

The following example script communicates with the LCS API to download the zip file for the RSAT certificate to the local machine. It shows the private certificate's password in the console window. An access token must be provided.
# Basic LCS API RSAT certificate zip download script

# This will download the RSAT certificate bundle for an environment to the current directory and display the private certificate's password in the console.
# The user used in the API authentication must be added to the project as an Environment Admin or Project Owner.

## Configuration
$accessToken = "{access token string}";
$projId = {project id integer};
$envId = "{environment id GUID}";
$baseLCSAPI = "lcsapi.lcs.dynamics.com";
$url = "https://$baseLCSAPI/environmentinfo/v1/rsatdownload/project/$projId/environment/$envId";
$headers = @{
    "Authorization" = "Bearer $accessToken"
    "x-ms-version" = "2017-09-15"
    "Content-Type" = "application/json"
}

# Reset variable between executions
$certificateResponse = $null
$shouldRetry = $false

do {
    $shouldRetry = $false
    try {
        # GET request to LCS API
        $certificateResponse = Invoke-RestMethod $url -Method 'GET' -Headers $headers
    } catch {
        # Check if this is a HTTP 429 error
        if ($_.Exception.Response.StatusCode.value__ -eq 429) {
            # Too many requests for this environment, wait and retry
            $shouldRetry = $true
            Write-Host "Too many requests - Retrying in $retrySeconds seconds"
            Start-Sleep -Seconds $retrySeconds
        } else {
            throw
        }
    }
} while($shouldRetry)

if ((-not $certificateResponse.IsSuccess) -or ($certificateResponse.Data -eq $null)) {
    Write-Host $certificateResponse.ErrorMessage
    throw
}

$fileName = $certificateResponse.Data.Filename

# Save the zip to the local disk.
# Could add unzipping in memory and install certificates to correct local certificate stores.
Set-Content $fileName -Value $certificateZip -Encoding Byte

Write-Host "Certificate bundle downloaded to $fileName with private certificate password $certificateSecret"
Rate limits

To better load balance requests, there are rate limits on this API. These limits are also shared with the LCS web interface.

- 1 call for each environment per minute

**NOTE**

Requests that exceed the rate limits will be rejected, and an "HTTP 429 Too Many Requests" response will be returned. The `retry-after` header will indicate the number of seconds that the request can be retried after.
Microsoft Dynamics Lifecycle Services (LCS) provides a cloud-based collaborative workspace that customers and their partners can use to manage projects from pre-sales through implementation and operations. LCS provides checklists and tools to help you manage your project, based on the phase of the project and the industry that you’re working in. It also provides a dashboard, so that you have a single location where you can obtain up-to-date project information.

To get started with LCS, see the Lifecycle Services (LCS) user guide.

LCS features and service changes will no longer be announced via blog posts. Descriptions of LCS features are provided in the release plans.

The following sections list the features that are included in LCS releases.

### June 2020 - wave 1

<table>
<thead>
<tr>
<th>AREA</th>
<th>FEATURE</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business process modeler (BPM)</td>
<td>Download task recording (AXTR)</td>
<td>General availability</td>
</tr>
<tr>
<td>Service updates</td>
<td>View canceled updates</td>
<td>General availability</td>
</tr>
</tbody>
</table>

### May 2020 - wave 2

This release contains general performance improvements and minor bug fixes.

### May 2020 - wave 1

<table>
<thead>
<tr>
<th>AREA</th>
<th>FEATURE</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue Search improvements</td>
<td>Lifecycle Services Issue search improvements</td>
<td>General availability</td>
</tr>
</tbody>
</table>

### April 2020 - wave 2

<table>
<thead>
<tr>
<th>AREA</th>
<th>FEATURE</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment actions</td>
<td>Lifecycle Services support for dual-write capabilities</td>
<td>Preview</td>
</tr>
<tr>
<td>Environment actions</td>
<td>Removing Remote Desktop access to Tier 2-5 Standard Acceptance Test (or sandbox) environments</td>
<td>Preview</td>
</tr>
</tbody>
</table>
April 2020 - wave 1
This release contains general performance improvements and minor bug fixes.

March 2020 - wave 2

<table>
<thead>
<tr>
<th>AREA</th>
<th>FEATURE</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin APIs</td>
<td>RESTful APIs for database export</td>
<td>Preview</td>
</tr>
<tr>
<td>Environment actions</td>
<td>Apply data upgrade packages for AX2012 customers on sandbox environments</td>
<td>General availability</td>
</tr>
<tr>
<td>Environment actions</td>
<td>Platform update 20 required for database movement operations</td>
<td>General availability</td>
</tr>
</tbody>
</table>

March 2020 - wave 1
This release contains general performance improvements and minor bug fixes.

February 2020 - wave 2

<table>
<thead>
<tr>
<th>AREA</th>
<th>FEATURE</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment actions</td>
<td>Platform update 20 required for database movement operations</td>
<td>General availability</td>
</tr>
</tbody>
</table>

February 2020 - wave 1

<table>
<thead>
<tr>
<th>AREA</th>
<th>FEATURE</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin APIs</td>
<td>Database movement RESTful APIs</td>
<td>General availability</td>
</tr>
</tbody>
</table>

LCS releases before November 2019
For information about LCS releases that occurred before November 2019, see the blog posts that the Lifecycle Services team published on the Dynamics 365 blog.
Microsoft Dynamics Lifecycle Services (LCS) provides regularly updated services. The goal of LCS is to deliver the right information, at the right time, to the right people, and to help ensure repeatable, predictable success with each roll-out of an implementation, update, or upgrade. LCS is available to customers and partners as part of their support plans. If you’re a customer of the newest version of the Dynamics 365 Finance and Operations apps, you can sign in by using your Microsoft Azure Active Directory (Azure AD) credentials. Go to LCS.

Tools that are provided in LCS

The following table lists the tools that are provided in LCS and describes the phases that each tool applies to.

<table>
<thead>
<tr>
<th>TOOL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projects</td>
<td>Projects are the key organizer for your experience in LCS. Projects let you invite your partners to collaborate with you, and they also let you track progress.</td>
</tr>
<tr>
<td>Methodologies</td>
<td>Methodologies provide a tool that you can use to ensure more repeatable, predictable implementation projects. You can use one of our methodologies or create your own. By using a methodology, you can easily track and report on your progress.</td>
</tr>
<tr>
<td>Business process modeler</td>
<td>Business process modeler lets you create, view, and modify standard process flows. By using Business process modeler, you can achieve the following goals: standardize process flows; align your business processes with industry-standard processes, as described by the American Productivity &amp; Quality Center (APQC); identify fit and gaps between user requirements and the default functionality that Microsoft Dynamics products provides.</td>
</tr>
<tr>
<td>Cloud-hosted environments</td>
<td>Cloud-hosted environments is a tool that you can use to deploy Microsoft Dynamics environments on Azure. When you use Cloud-hosted environments, you must select the type of environment to deploy, such as a demo, developer/test, or production environment. Based on your selection, the Cloud-hosted environments tool provisions the appropriate number of virtual machines (VMs) in Azure. These VMs have Microsoft Dynamics components (and all their prerequisites) already installed on them.</td>
</tr>
<tr>
<td>Cloud-powered support</td>
<td>Cloud-powered support helps you manage support incidents. It lets you create a VM in Azure that has the same hotfixes installed as your local environment. You can reproduce and record an incident on the VM, and then submit the incident to our support team. Support follows up by investigating and, if possible, testing a fix on the VM, and then sends the fix back to you for verification.</td>
</tr>
<tr>
<td>TOOL</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>Configuration and data manager (preview)</td>
<td>Configuration and data manager (preview) lets you copy a configuration from one instance to another. You can copy from and to environments that meet the following criteria: they are managed as part of an LCS project; they run the Data Import/Export Framework.</td>
</tr>
<tr>
<td>Customization analysis</td>
<td>Customization analysis validates model files against best practices and provides a report of potential areas for improvement.</td>
</tr>
<tr>
<td>Issue search</td>
<td>Issue search helps you find existing solutions and workarounds for known issues in Microsoft Dynamics products. You can see which issues have been fixed, which issues remain open, and which issues have been resolved as &quot;won't fix.&quot;</td>
</tr>
<tr>
<td>Asset library</td>
<td>The Asset library is a storage location for the various assets that are associated with a tenant in LCS.</td>
</tr>
<tr>
<td>Get updates</td>
<td>Get updates is a tool that customers to access the updates that are available for their environments.</td>
</tr>
<tr>
<td>Environment monitoring</td>
<td>Environment monitoring is a set of tools that helps you monitor, diagnose, and analyze the health of the environments that you manage.</td>
</tr>
<tr>
<td>Translation service</td>
<td>The Microsoft Dynamics 365 Translation Service (DTS) is hosted in LCS. It’s designed to enhance the experience for partners and independent software vendors (ISVs) when they translate their solutions or add a new language for the supported Dynamics products.</td>
</tr>
<tr>
<td>Regulatory updates</td>
<td>Regulatory updates is a tool that lets customers, partners, and ISV solution providers stay up to date about regulatory updates by setting up alerts through LCS.</td>
</tr>
<tr>
<td>System diagnostics</td>
<td>System diagnostics is a tool that helps administrators monitor AX 2012 environments.</td>
</tr>
<tr>
<td>Updates</td>
<td>The Updates page hosts the details of updates that are available for an AX 2012 environment. It also provides access to groups of updates that can be used for slipstream installations.</td>
</tr>
<tr>
<td>Upgrade analysis</td>
<td>Upgrade analysis helps you plan your upgrade to the latest version by analyzing code artifacts from Microsoft Dynamics AX 4.0, Microsoft Dynamics AX 2009, or AX 2012.</td>
</tr>
<tr>
<td>Usage profiler</td>
<td>Usage profiler is a data-gathering tool that helps you describe your projected or current usage of an. The usage profile that is generated can be used for various purposes, such as hardware sizing and support.</td>
</tr>
<tr>
<td>Downloadable tools</td>
<td>The Downloadable tools page provides a set of tools that were previously hosted on Microsoft Dynamics InformationSource.</td>
</tr>
</tbody>
</table>
License sizing estimator helps you estimate the number of licenses that are required. It provides a shared workspace that lets you model default and customized roles, and then automatically calculate the required client access licenses (CALs).

## Additional resources

The LCS team is also blogging on the [Lifecycle Services Engineering blog](https://lifecycleservicesengineering.com). Subscribe to our RSS feed to keep up with our posts and announcements.

**Access Lifecycle Services**
Project onboarding is a self-paced, wizard-driven onboarding experience that guides project users in Microsoft Dynamics Lifecycle Services (LCS) through the process of setting up the key configuration components for a new implementation project for Dynamic 365 Finance, Dynamics 365 Supply Chain Management, or Dynamic 365 Retail. This wizard can also be accessed during and after the implementation, and can be used to update the information as required.

Microsoft relies on the information that you provide. You must provide the most current and accurate data as you complete Project onboarding. After you complete Project onboarding, you can deploy environments and continue with the project implementation.

To access Project onboarding, sign in to LCS, and then, on the main menu, select Project onboarding.

**NOTE**

Project onboarding is available for implementation projects and must be completed before any of the Microsoft-managed environments are deployed. For more information about implementation projects, see Lifecycle Services (LCS) for Finance and Operations apps customers.

For more information about the onboarding process, see Onboard an implementation project, and watch the Finance and Operations: Onboarding to Dynamics 365 TechTalk.

### Onboarding steps

Each step in Project onboarding is designed to give you guidance about the project implementation or to gather information about the project context, so that the FastTrack team can better serve you. By providing accurate information in the wizard, you help Microsoft understand your implementation plan, so that it can provide appropriate guidance.

You can move through the wizard either by using the Next and Back buttons, or by selecting each step directly. For some steps, the right column of the page shows additional contextual information that will help you complete the step.

### Welcome

The Welcome page provides general guidance and information that you will need to complete Project onboarding.

### Project overview

- Provide the overview information for the implementation project.
- Describe the vision and goals for the project in a few sentences. This information will help Microsoft understand the goals that you want to achieve and how you define success for the project.
- Provide the Partner MPN ID, which you can get from the implementation partner team. If a partner is not involved or not yet identified, choose the appropriate option in the implementation partner drop-down list. Note that providing accurate partner data is a pre-requisite for FastTrack Program assignment. You could miss the opportunity for valuable services if you do not provide the correct partner information. Once partner identified, you need to update the MPN ID.
• Specify the estimated number of user licenses after full roll-out including current licenses. This number can differ from the current license purchase. If no change is planned, provide the current user license count. If a license type isn’t applicable, enter 0 (zero).

• If your implementation project is a demo project, or if you’re moving from another tenant, provide the details.

Project scope

• Provide information about how you plan to scope the implementation in terms of features and products. This information will help Microsoft understand your implementation and provide any guidance that is required.

• After you set the option for specific features to Yes in this step, you must provide additional data that is related to those features. This data will help Microsoft have a better engagement with you during the implementation. The additional data fields are mandatory.

Define your team

• Verify that all project team members are invited and configured.

• Set the Primary contact for FastTrack option to Yes for at least two users who have an active email address in the user list. If this option isn’t set to Yes for any team member, FastTrack will reach out to all team members for implementation guidance during your implementation. If necessary, you should nominate at least one customer and one partner team member to be contacted by FastTrack.

• Each team member will be assigned a project security role and an implementation role. The project security role is relevant to access to the LCS project workspace, and the implementation role is relevant to the individual team member’s role on the implementation team. We highly recommend that you include representatives from the customer among the project team members who have a monitored email address.

For more information, see Configuring project security and Roles in a Dynamics 365 implementation.

Define milestone dates

• Define all mandatory milestone steps. Milestones are associated with the methodology of the project. If the milestone dates haven’t yet been decided, use tentative dates.

• Update the milestone dates as plans change. Microsoft uses the milestone dates to provide appropriate guidance for each milestone. To edit milestone dates, select the pencil symbol.

Associate LCS with Azure DevOps

• Connect LCS and Azure DevOps to maintain the application lifecycle.

• Enter the root URL of your Azure DevOps account and the personal access token that you obtained from Azure DevOps. The Azure DevOps account should belong to the customer.

Configure Azure DevOps

• Map work items between LCS and Business process modeler (BPM).

• Acknowledge the setup.

• If you choose to use a custom process template, follow these best practices:
  ○ Don’t remove any existing work item types.
  ○ Don’t remove any existing state for a work item type.
  ○ Don’t add any required fields to a work item type.
FastTrack

- The FastTrack page introduces the FastTrack program. It explains what the program consists of and how your implementation can benefit from it.
- We strongly recommend that you watch the existing TechTalks and subscribe to upcoming TechTalks as Microsoft continues to share best practice guidance and information about the changes that are occurring in the product and platform.

Next steps

The Next Steps page provides additional resources about the most critical aspects of the implementation. You can access this page at any time during the implementation.

Complete onboarding

- Complete Project onboarding so that you can move on to the next steps in your implementation.
- You can complete Project onboarding only after all previous steps have been completed. If any previous steps haven't been completed, the Complete onboarding button won't be available.
- Any skipped steps are marked with an asterisk (*). You can also view any missing steps.
- After you complete Project onboarding, you can continue to update information, such as project scope values.
Subscription estimator is a tool that is available in Microsoft Dynamics Lifecycle Services (LCS). Microsoft uses this tool to estimate the initial size of the production environment that must be provisioned for a customer. Before customers can request deployment of a production environment, they must estimate their peak workloads in terms of transaction counts and then upload that information to LCS. By using the details of user licenses and transaction counts to infer subscription requirements, the Subscription estimator tool helps ensure that the provisioned environment meets the customer’s business requirements.

Follow these steps to use the Subscription estimator tool.

1. In LCS, open the project that is associated with the implementation project.

2. At the top of the page, select the hamburger icon, and then select Subscription estimator.

3. Download the sample usage profile.

4. Answer the required questions on each tab. If you're a Commerce customer, be sure to answer the questions on the Retail and Commerce tab.

5. Save the usage profile locally.

6. To upload the usage profile, select New estimate, name the estimate, and then upload the usage profile.

7. After the upload is completed, select Mark as Active to activate an estimate. An active estimate is required in order to configure a production deployment.

When there is a valid active estimate, the Configure button becomes available. You can use this button to request a production environment deployment.
Frequently asked questions

Why isn't the Configure button for deploying a production environment available, even though there is an active estimate? And why does a warning message appear in the Action Center on the project dashboard?

If you have multiple implementation projects, the Configure button might not be enabled and a warning message will appear in the Action Center regarding an insufficient number of licenses. Log a support request, and the Support team can help resolve this issue.

Why does an error occur when I mark an estimate as Active?

When you mark an estimate as Active, you might receive the following error message:

Estimate created but does not meet requirements

This error occurs if transaction lines that are entered aren't within the limits of the Subscription estimation tool. To resolve this error, create a support request, and attach the usage profile. Your instance can then be manually sized.

How can I update my subscription if my production environment is deployed?

The Subscription estimator is a required step before requesting production. Although you can have multiple estimates, one must be marked as Active. The active subscription estimate is used to size the production environment. After the production environment has been deployed, or deployment of the environment has received sign-off, the active estimate is locked. To mark a different estimate as the active estimate, create a support request by using the Support portal in LCS. If you have an increase in the licensing volume of your project, you will need to create a new subscription estimator and log a support request to have it marked as Active. Resizing may be applicable based on the new subscription estimate.

What should I do to activate my subscription estimate if I have multiple projects in the same tenant?

When you are implementing several projects in the same tenant, a warning indicating "subscription estimate is not complete" may appear in the Action Center of LCS. This error will indicate that the total number of estimated users for all implementation projects should not exceed the number of purchased licenses. This may happen if the sum of users on the active subscription estimates is superior to the tenant license count of the same type. To have this corrected, you should submit a support request to Microsoft as soon as possible, asking for the subscription estimations to be corrected, including the information about license allocation. With the help of the Support team, the process for the estimation edition will be enabled.

Make sure that you have all the required licenses active before submitting the request and be aware that in cases were resizing of the production environment is needed, it may require downtime.

NOTE

Although you can have multiple estimates, one estimate must be marked as Active. After the production environment has been deployed, or deployment of the environment has received sign-off, the active estimate is locked. To mark a different estimate as the active estimate, create a support request by using the Support portal in LCS.

NOTE

FastTrack Solutions architects have no involvement in uploading or updating the Subscription Estimator. If you identify any warnings regarding the Subscription Estimator in LCS, follow the instructions above. If you continue to have issues, contact Microsoft Support.

If you receive any other error message or encounter any other issue, create a support request, and attach your active estimate so that the Support team can address the issue.
Additional resources

Subscriptions, LCS projects, and Azure Active Directory tenants FAQ
Security in Microsoft Dynamics Lifecycle Services (LCS) is controlled at both the organization level and the project level. Not all members of an organization have access to all projects. Additionally, the members of a project might not all be members of the same organization.

Currently, users can sign in by using the Microsoft Azure Active Directory (Azure AD) credentials that they created in the Microsoft 365 portal when they signed up. Users who are administrators for their organization in Azure AD will be administrators in Lifecycle Services (LCS).

Project-level access to LCS is by invitation. You can invite members of your organization to be project owners and team members. Additionally, you can invite users who aren’t part of your organization, and who don’t have accounts in Azure AD to be team members.

**IMPORTANT**
We strongly recommend that you manage all users within your company at the organization level. Additionally, you help ensure that users can access the benefits that are available to your organization.

### Manage LCS organization users

Only an administrator can manage users. Follow these steps.

1. In PartnerSource Business Center (PSBC) or Azure AD, associate all the users in your organization who require access to LCS with your organization. Users might have to wait up to two business days before they can sign in to LCS.
2. Add your users to the appropriate projects in LCS.

### Invite a user to an LCS project

1. Sign in to LCS.
2. Select the project to add the user to.
3. Select the **Project users** tile, and then, on the **Project users** page, select the plus sign (+).
4. Enter the user’s email address, select the correct security role, and then select **Invite**.

**NOTE**
For implementation projects, you can select the implementation role for the invited user. If you set **Allow FastTrack to contact** to **Yes**, then Microsoft FastTrack team may reach out to you based on your implementation role and the stages of the implementation project.

### Configuring project security

You can invite users from inside or outside your organization to join your project as users. The following table describes the roles that are available for users.

<table>
<thead>
<tr>
<th>ROLE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>ROLE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Project owner</td>
<td>Members of this role have access to all tools in LCS, can add other users in any role, and can delete the project.</td>
</tr>
<tr>
<td>Environment manager</td>
<td>Members of this role have access to all tools in LCS and can manage cloud-hosted environments.</td>
</tr>
<tr>
<td>Project team member</td>
<td>Members of this role have access to all tools in LCS but can't manage cloud-hosted environments.</td>
</tr>
<tr>
<td>Project team member (prospect)</td>
<td>Members of this role have limited access to all tools in an LCS project. Prospects are users who have been added to a project, but who don't have an account in VOICE or an Azure AD account. You can identify that a user is a prospect, because <strong>prospect</strong> is listed as the organization.</td>
</tr>
</tbody>
</table>
| Operations user                           | Members of this role have access to the following tools in LCS:  
  - System diagnostics  
  - Issue search  
  - Cloud-powered support  
  - Updates  
  - Cloud-hosted environments |

After you’ve configured security for one project, you can import the users to another project.

**Configure implementation roles**

If you have an implementation project, you will have the option to specify project user’s implementation roles. For more information, see [Roles in a Dynamics 365 implementation](#).
This article provides information about the Issue search tool on Microsoft Dynamics Lifecycle Services (LCS). It explains how to search for product issues and regulatory features, and describes the information that is provided for each status.

Prerequisites
None

Search for product issues and regulatory features

You can use Issue search to search for product issues, and determine whether an issue has been resolved, is open, or has a workaround. You can also search for regulatory features, and determine whether a feature is available or is planned in a future release. Finally, you can find regulatory white papers, certifications, and registrations.

1. Go to Microsoft Dynamics Lifecycle Services (LCS).
2. Select a project to work in.
3. Click the Issue search tile.
4. Enter search terms. You can enter a keyword or group of keywords, or a Microsoft Knowledge Base (KB) number. You can also use a dollar sign ($) to indicate an Application Object Tree (AOT) object path in the format $\{ObjectType\}Object or $\{ObjectType\}Object#Method (for example, $\{Classes\}Tax#Save). Standard search operators such as AND and OR are supported.

You can filter the results list for resolved or open issues, workarounds, and issues that won’t be fixed or are postponed. You can also filter by the application version. By default, the results are sorted by relevance. However, you can sort by date ascending, date descending, version ascending, or version descending instead. By default, all status and product version filters are selected. Note: Results for Microsoft Dynamics AX 2009 are included in Microsoft Dynamics AX 2012 projects only when you search for regulatory features. The following table describes the information that is provided for each status when you search by product issue.

<table>
<thead>
<tr>
<th>STATUS</th>
<th>COLOR</th>
<th>DESCRIPTION OF RESULTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STATUS</td>
<td>COLOR</td>
<td>DESCRIPTION OF RESULTS</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Resolved    | Green    | The KB number, title, version, description of the problem and change, and hotfix are provided. A list of objects that are affected by the hotfix is also provided whenever possible.  
  - To download the hotfix, click Download hotfix.  
  - To determine what code was changed in a hotfix, click View changes. Added code is indicated by a green highlight, and deleted code is indicated by a red highlight. Important: The list of code changes is provided for reference only. Code changes should not be manually inserted into a higher development layer. Otherwise, they become unsupported customized objects that the partner or customer assumes full responsibility for. |
| Open        | Red      | A bug number, title, version, and description of the problem are provided.                                                                                                                                       |
| Workarounds | Brown    | The issue number, title, version, problem description, and mitigation are provided.                                                                                                                             |
| Postponed   | Gray     | A bug number, title, version, and description of the problem are provided. If a workaround is available, it's described. A bug that has this status has been evaluated and won't be fixed at this time. |
| By design   | Gray     | A bug number, title, version, and description of the problem are provided. An explanation of the design is provided whenever possible. A bug that has this status has been evaluated, and it has been determined that this functionality is working as designed. |
| Not reproducible | Gray | A bug number, title, version, and description of the problem are provided. An explanation of the problem is provided whenever possible. A bug that has this status has been evaluated, and a fix won't be issued at this time. |
Will not be fixed

A bug number, title, version, and description of the problem are provided. If a workaround is available, it’s described. A bug that has this status has been evaluated and won’t be fixed.

Resolved

For feature updates: The KB number, title, version, description of the problem, and hotfix are provided. A list of objects that are affected by the hotfix is also provided whenever possible.
- To download the hotfix, click Download hotfix.
- To read the KB article for the hotfix, click Read KB article.

For white papers, certifications, registrations, and reports: The title, product version, and description of the white paper, certification, registration, or report are provided.
- To read the documentation, click Read documentation.

Open

A bug number, title, version, planned release date, and description of the problem are provided.

The following table describes the information that is provided for each status when you search by regulatory feature.

<table>
<thead>
<tr>
<th>STATUS</th>
<th>COLOR</th>
<th>DESCRIPTION OF RESULTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Will not be fixed</td>
<td>Gray</td>
<td>A bug number, title, version, and description of the problem are provided. If a workaround is available, it’s described. A bug that has this status has been evaluated and won’t be fixed.</td>
</tr>
</tbody>
</table>
| Resolved       | Green | For feature updates: The KB number, title, version, description of the problem, and hotfix are provided. A list of objects that are affected by the hotfix is also provided whenever possible.  
- To download the hotfix, click Download hotfix.  
- To read the KB article for the hotfix, click Read KB article.  

For white papers, certifications, registrations, and reports: The title, product version, and description of the white paper, certification, registration, or report are provided.  
- To read the documentation, click Read documentation. |
| Open           | Red   | A bug number, title, version, planned release date, and description of the problem are provided. |

Issue details

The code changes are provided for reference only and should not be manually inserted into a higher development layer. Otherwise, they become unsupported customized objects that the partner or customer assumes full responsibility for.
You can use the Configuration manager to copy from and to Dynamics AX 2012 R3 environments that meet the following criteria:

- Managed as part of a Lifecycle Services project
- Running System diagnostics
- Running the Data Import/Export Framework

**IMPORTANT**

This feature is not supported for production use. Configuration manager (beta) relies on entities from the Data Import/Export Framework in your environment. Because these entities do not currently include all the functionality in AX 2012 R3, some configuration data is not copied between environments.

For more information, see:

- Set up Configuration manager
- Copy configurations by using Configuration manager
Set up Configuration manager

11/24/2021 • 3 minutes to read • Edit Online

IMPORTANT
This feature is not supported for production use. Configuration manager (beta) relies on entities from the Data Import/Export Framework in your environment. Because these entities do not currently include all the functionality in AX 2012 R3, some configuration data is not copied between environments.

Before you begin

Before you begin, your environment must include the following components:

- A running version of AX 2012 R3 that has been configured for your business. For more information about how to install AX 2012 R3, see Install Microsoft Dynamics AX 2012.

- A running instance of the Data Import/Export Framework. For more information about how to install the Data Import/Export Framework, see Install the Data import/export framework (AX 2012 R#).

- An AX 2012 R3 project in Lifecycle Services.

WARNING
Copying configurations between environments can be a destructive operation. All project owners have the right to configure and perform these operations. Make sure that only trusted individuals are set as project owners.

Create Data Import/Export Framework source data formats in AX 2012 R3

You must create both a Dynamics AX and a CSV source data format to manage configurations in all environments where you intend to export and import configurations.

Create an AX source data format

Complete the following procedure in the environment that you intend to export a configuration from.

1. Click Data import export framework > Setup > Source data formats.
2. Click New, and name the source AX.
3. Verify that the type is AX.
4. Repeat this procedure in the environment that you intend to import the configuration to.

Create a CSV source data format

Complete the following procedure in the environment that you intend to export a configuration from.

1. Click Data import export framework > Setup > Source data formats.
2. Click **New**, and name the source CSV.

3. Verify that the type is **File**.

4. On the **General** tab, set the following values.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>File format</td>
<td>Delimited</td>
</tr>
<tr>
<td>First row header</td>
<td>Selected</td>
</tr>
<tr>
<td>Row delimiter</td>
<td>(CR)(LF)</td>
</tr>
<tr>
<td>Column delimiter</td>
<td>Vertical bar (</td>
</tr>
<tr>
<td>Text qualifier</td>
<td>@@</td>
</tr>
<tr>
<td>Skip row</td>
<td>0</td>
</tr>
<tr>
<td>Code page</td>
<td>1252</td>
</tr>
<tr>
<td>Language locale</td>
<td>EN-US</td>
</tr>
<tr>
<td>Multiple value separator</td>
<td>;</td>
</tr>
</tbody>
</table>

5. Repeat this procedure in the environment that you intend to import the configuration to.

**Install and configure the local component of the System diagnostics (Lifecycle Services)**

Complete the following procedure in the environment that you intend to export a configuration from.
1. Install the local component of the System diagnostics. For details, see Install and run System diagnostics. Important: For this beta release, we require that you add the service account for the System diagnostics to the sysadmin role in AX 2012 R3.

2. Click Start > Microsoft Dynamics AX Lifecycle Services Diagnostic Service Discovery.

3. In the Environment Discovery window, enter a name for the environment, and the fully-qualified name of the Microsoft SQL Server instance and database. Then click Discover environment.

4. After discovery is completed, enter the values in the Configuration management (Beta) section, click Save, and then click Upload environment.

<table>
<thead>
<tr>
<th>OPTION</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable configuration overwrites</td>
<td>Select this option to enable configurations to be copied and overwritten for the specified AOS instance. Important: We strongly recommend that you disable configuration overwrites when you have fully configured an environment.</td>
</tr>
<tr>
<td>AOS instance</td>
<td>Specify the AOS instance that can be copied from or overwritten.</td>
</tr>
<tr>
<td>Storage location</td>
<td>Specify the location where configurations are stored locally. The AOS service account and the Data Import/Export Framework service account must have read and write access to this location. Important: We strongly recommend that you use the same directory that is used for the Data Import/Export Framework. Be aware that the shared directory might contain sensitive data, depending on what you are importing and exporting. Make sure that as few users as possible, in addition to the AOS service account and the Data Import/Export Framework service account, have access to the location.</td>
</tr>
</tbody>
</table>
5. Repeat this procedure in the environment that you intend to import a configuration to.

Next steps

The environment is now ready for you to copy and manage configurations. For more information, see Copy configurations by using Configuration manager.
Configure the code upgrade service in Lifecycle Services (LCS)

11/24/2021 • 5 minutes to read • Edit Online

This topic explains how to configure the **Code upgrade** tile in Lifecycle Services (LCS) to migrate your solution to the latest version of the Dynamics 365 Finance and Operations apps.

**Overview**

The code upgrade tool operates by connecting to your Azure DevOps project, locating your Trunk\Main branch, branching to a new branch that will be named as Releases\<version number>, and then performing the code upgrade there. After this process is complete, you can synchronize your developer environment to this new branch under Releases\<version number> and resolve conflicts. When you have compiled and tested your upgraded code you can merge the new branch back into Trunk\Main, using source control explorer in Visual Studio and the process is complete.

Dynamics 365 for Finance and Operations version 8.0 and newer, does not allow customization via overlayering of Microsoft models. Before you upgrade, you must have a plan to refactor your customizations into extensions. For more information, see the [Extensibility home page](#) and [Relax model restrictions to refactor overlayering into extensions](#).

**Process**

**Create the Trunk\Main folder structure**

For the code upgrade service to recognize your source code, your Azure DevOps project must contain a Team Foundation Version Control (TFVC) code repository. In addition, the code repository folder structure must conform to the following strict pattern.

- For code and metadata: `/<DevOps project name>/Trunk/Main/Metadata`
- For Visual Studio project and solution files: `/<DevOps project name>/Trunk/Main/Projects`

You can create new folders directly in the Azure DevOps web interface under **Repos**.

**NOTE**

- Folder names are case sensitive, that is, you must use Main and not MAIN, or the code upgrade service will not recognize the folder.
- Azure DevOps projects use Git version control by default. You will need to add a TFVC repository.
  1. Go to Project settings, then Repositories.
  2. Select New repository.
  3. In the Type field, select TFVC, and then click Create.

**Create a personal access token**

To connect to an Azure DevOps project, LCS is authenticated using a personal access token. Use the following steps to create a personal access token in Azure DevOps. If you have already configured your LCS project to connect to your Azure DevOps project, you can skip this section.

1. Sign in to visualstudio.com and locate your Azure DevOps project.
2. In the top-right corner, hover over your name, a menu appears, select **Security**.

3. Select **Add** to create a new personal access token, give it a name, and then enter the amount of time that you want the token to last for. Select **Create Token**.

4. Copy the token to your clipboard. You will not be able to find the token details after this step is completed, so be sure that you have copied the token before navigating away from this page.

**Configure your Lifecycle Services project to connect to Azure DevOps**

1. In your LCS project, go to the **Project settings** tile, select **Visual Studio Team Services**, and then select the **Setup Visual Studio Team Services** button. This configuration is needed by many LCS tools, if you have already configured LCS to connect to your Azure DevOps project, you can skip this section.
2. Enter the root URL for your Azure DevOps organization and the access token created earlier, and then select Continue.

3. Select the project within your Azure DevOps organization that you want to connect to, and select Continue.

4. On the Review and save page, select Save.

Create an ax7.version file

NOTE
If you are migrating from AX 2012, you can skip this step.
The code upgrade tile in LCS automatically finds the version that you are migrating from, by reading the `ax7.version` file under the Main folder in your source control. You must create this file manually, either in Visual Studio or through the Azure DevOps web portal, as shown below. This file is not needed if you are migrating your code from Dynamics AX 2012 R3 or an earlier version. The version number entered here must be the application version (not the platform version). Take care to enter the correct version number here as entering an incorrect version number in this file may cause your code upgrade run to fail.

For more information about how to identify which application version you have, see [Overview of Microsoft Dynamics AX build numbers](#).

**Execute the code upgrade tile**

1. In your LCS project, select the **Code upgrade** tile.

2. In the bottom-left corner of the screen, select **Add**, and then enter a name and description. Select the version you are upgrading from as Microsoft Dynamics AX 7, and then select **Create**.

   - If you are upgrading your code from Dynamics AX 2012 R3, select the version you are upgrading from. You will be prompted to upload a zipped version of your Dynamics AX 2012 R3 model store file.
   - If the **Estimation Only** check box is selected, the tool only generates a report and does not check in or create a new code branch in Azure DevOps for you. You should use this option if you want to evaluate the potential size of the work involved in upgrading before you commit to the actual upgrade.
3. Select **Analyze code** in the bottom-right corner. The code upgrade process will start. This process typically takes 40 minutes for a large solution to complete. When complete, return to the **Code upgrade** tile in LCS to view the results.

4. The code upgrade service creates a new branch and checks in the upgraded code to your Azure DevOps project. After the upgrade process is complete, your code will exist in a new branch under the **Releases** folder. The branch name is suffixed with the date and time of the upgrade.

**Merge Releases back into Trunk\Main**

Once the upgraded code in **Releases\<version number>** compiles successfully and you have completed your code migration and testing, you are ready to merge this branch back into **Trunk\Main**. To merge, on your development environment in Visual Studio, open the Source control explorer pane then right-click on the **Releases\<version number>** branch, and in the context menu go to **Branching and Merging**, and then on the submenu select **Merge**.
The Source Control Merge Wizard opens, which guides you through merging the Releases\<version number> branch back into Trunk\Main.
Lifecycle Services (LCS) for Microsoft Dynamics provides methodologies that you can use to ensure a more repeatable and predictable implementation project experience. You can use one of the provided methodologies or create your own. With a methodology, you can easily track and report on your progress.

A methodology consists of phases, tasks, and milestones. Each phase can have any number of tasks, some of which are mandatory. When all of the tasks in a phase are completed, the phase can be marked as complete. You can also create a milestone for when you anticipate a phase to be completed. The following methodologies are included in an LCS project:

- Implementation
- Sure Step
- Learn development
- Migrate and create solutions
- Consume solutions

Note: There is a known limitation where new changes that are published to the Microsoft Methodology are not pushed to existing projects. Only new projects get these changes.

Add or update methodologies

A partner or a project administrator can create new methodologies or make changes to an existing methodology for their organization or within the scope of a specific project. These additions and changes can be made at the project level or at the organization level. Use the following procedures to create and save a new methodology, update existing methodologies, and when appropriate, promote a new methodology or methodology changes to the organization level.

Create a new methodology

1. On the Lifecycle Services dashboard, on the right side of the screen, click Manage methodologies.
2. On the **Manage methodologies** page, click the plus sign (+).
3. In the New methodology pane, enter a name and description for the new methodology. Click Confirm.
4. Optional: After you have confirmed the methodology, you can promote it to the organization level by selecting the methodology in the grid, and then selecting Promote.
Change or update a methodology

There are two ways to make changes to a methodology. You can append an existing methodology or you can make changes to a methodology in the scope of a project. From the LCS project dashboard, select the methodology that you want to update, and then select Edit methodology or Append methodology.

If you select to edit the methodology, you can make the following changes:

- Add a new phase.
- Add a new task.
- Edit a phase or task. (Some phases and tasks can be edited, but others are enforced by Microsoft and are therefore locked and can't be edited.)
- Copy a phase or task.
- Reorder phases and tasks.
- Delete a phase or task.
Business process modeler (BPM) in Microsoft Dynamics Lifecycle Services (LCS) is a tool that you can use to create, view, and modify repeatable implementations that are based on business process libraries. BPM helps you align your business processes with industry-standard processes that are described by the American Productivity & Quality Center (APQC). You can perform fit-gap analysis between your business requirements and the default processes in Finance and Operations apps. Additionally, you can add new business processes that aren’t already defined.

BPM is compatible with the following products:

- **Microsoft Word** – You can generate documentation for business processes.
- **Microsoft Visio** – You can export business process maps to Visio files.

**NOTE**

The information in this topic is specific only to Finance and Operations apps. For information about Business process modeler and Microsoft Dynamics AX 2012, see Business process modeler (BPM) in Lifecycle Services (LCS).

**Prerequisites**

To effectively use BPM, you must have Microsoft Office 2010 and a Microsoft Azure DevOps project.

**Getting started**

Follow these steps to access BPM.

1. Go to LCS.
2. Sign in, open a project, and then select the **Business process modeler** from the drop-down menu. The **Business process libraries** page has three sections:
   - **Project libraries** – This section contains business processes that a user has created or added.
   - **Corporate libraries** – This section contains custom business processes that someone in your organization has published.
   - **Global libraries** – This section contains cross-industry standard business processes, typically published by Microsoft.
3. To copy a standard business process library from the **Global libraries** section to the **Project libraries** section, select the upper-right corner of the tile in the **Global libraries** section, and then select **Copy**.
4. After the business process library has been added to the **Project libraries** section, select the tile to view the business process library.

**Additional resources**

- Create, edit, and browse Business process modeler (BPM) libraries
- Synchronize BPM libraries with Azure DevOps
- Complete tasks in Business process modeler (BPM)
● Work with activity diagrams in Business process modeler libraries
● Business process libraries in Business process modeler (BPM)
● Flowcharts in Business process modeler (BPM)
This article explains how to view a standard business process library in Business process modeler, how to copy and modify a business process library, and how to export information about the business process library to Microsoft Word.

This topic explains how to view a standard business process library, how to copy a business process library, modify it, and how to export information for the business process library to Microsoft Word.

View and copy a standard business process library

You can select a standard business process library on the Business process library page. To open this page, sign in to Lifecycle Services, open a project, and then click the Business process modeler tile.

To select a standard business process library to start with, follow these steps:

1. Sign in to Lifecycle Services, open a project, and then click the Business process modeler tile.
2. In the Global libraries or Corporate libraries section, right-click a library.
3. On the app bar, click Copy. The library is added to the My libraries section.
4. In the My libraries section, click the library to display the business process library.

Search for a process within a library

You can search for a relevant business process within a business process library.

1. Sign in to Lifecycle Services, open a project, and then click the Business process modeler tile.
2. Open a business process library.
3. In the search field, enter a search term or phrase, or a $ followed by the AOT name of an object. For example, $LedgerJournalTransDaily.

Modify a business process library

You can modify a business process library if it is associated with a project as described in the previous procedure. To modify a business process library, follow these steps:

1. Sign in to Lifecycle Services, open a project, and then click the Business process modeler tile.
2. Open a business process library.
3. Make changes to the business process library.
   - To change an existing library node, right-click the node to display the app bar, and then click Edit. Make changes, and then click Save.
   - To add a library node, drag a node from the Activities list to the library. Right-click the new node and then click Edit to change the name and other information for the node. Make changes, and then click
Save.

- To delete a library node, right-click the node, and then click **Delete**.

**Export a business process library to Word**

You can export information about a business process library, and all the flowcharts that are associated with it, to a Word document. To export a business process library to Word, follow these steps:

1. Sign in to Lifecycle Services, open a project, and then click the **Business process modeler** tile.
2. Open a business process library.
3. In the **Core Business Processes** list, right-click a top-level node in the library.
4. On the app bar, click **Doc**, and save the document.
This topic provides information about how to create, edit, and browse Business process modeler (BPM) libraries. It's important to note that you can browse a BPM library that is a global library or a corporate library. However, before you can edit and work with a BPM library, it must be part of your project in Microsoft Dynamics Lifecycle Services (LCS). Libraries that are distributed by Microsoft appear under **Global libraries**, whereas libraries that are published by your organization appear under **Corporate libraries**.

**NOTE**

BPM localization is not supported. If you edit in the new BPM client in any language other than EN-US, your changes will only display when you view the BPM in the language in which the changes were made. To view any changes made in EN-US, you must synchronize with Visual Studio Team Server before the changes will display.

Create a BPM library

There are several ways to author a BPM library. You can do so from scratch either building directly in the client or by importing an Excel template. Additionally, you can copy an existing library. This section walks through each of these methods.

**Use the BPM client**

1. On the **Business process libraries** page, select **New library**.

   **Business process libraries**

   🔄 New library  📑 Import from Excel

   **Project libraries**

   📑 (August 2016) APQC Unified Library for Microsoft Dynamics AX
   🔄 Microsoft Dynamics AX
   📑 Microsoft Dynamics AX

   📗 Microsoft Approved

   ...  ...

   2. Enter a name for the new library, and then select **Create**.
Use Excel Import

1. On the Business process libraries page, select Import from Excel.

2. Select Download template from the pane. Once downloaded, open the file.

3. The template has several columns, most importantly Id and Parent Id. Associate each line with a new Id number, if you’d like to make a line a child item, add the Id of the line you’d like it to fall under in the parent Id column. the

4. Once complete, save the template and return to BPM.

5. Using the import pane, select Browse to upload the updated template, enter a name for the new library, and select Import.

Copy a library

1. Open the Business process libraries page.

2. On the tile for the library that you want to copy, select the ellipsis button (…), and then select Copy.
Corporate libraries

3. Enter a name for the library, and then click **Create**.

**Import a sections of another library**

1. Open the **Business process libraries** page, and then open the library you want to edit.
2. Navigate to the line you would like to import to and select **Import**.
Add a new process

1. In the BPM library, select an existing process.

2. Select **Add process**. You can select to add the process as a child or a sibling of the selected process node. In this way, you can create a semantic hierarchy of business processes.

Edit the properties of a process

1. In the BPM library, select the process node to edit.

2. In the right pane, on the **Overview** tab, click **Edit mode**.
3. Enter a name and description for the process node.

4. Select the industries and the countries or regions that the process applies to. You can also add keywords and links. Keywords let you define categories, work streams, or other metadata. Links (URLs) let you reference external sites or documentation.

5. When you’ve finished editing the properties, click Save.

**Move a process**

You can move a process node or assign it to another parent node in the BPM hierarchy.

1. Select the process node to move, and then click **Move process**. You can select to move the process up or down, or you can select **Move** to see more options.

2. If you selected **Move**, you can browse the hierarchy, select a node to move the process to, and then select **Move as child** or **Move as sibling**. To cancel the move operation, click **Cancel**.

**Delete a process**

To delete a business process, select the process to delete, and then select **Delete**.

**Copy a global or corporate library to your project**

You can browse a BPM library that is a global library or a corporate library. However, before you can edit and work with a BPM library, it must be part of your project in Microsoft Dynamics Lifecycle Services (LCS). Libraries that are distributed by Microsoft appear under **Global libraries**, whereas libraries that are published by your organization appear under **Corporate libraries**.

**Browse a BPM library**

1. On the **Business process libraries** page, double-click the tile for the library that you want to browse.

2. In the BPM library, select a process to view its substeps.
3. Use the buttons on the toolbar to add, delete, or import processes as a child or a sibling. You can also select **Collapse all** to view only parent processes.

### Search a BPM library

You can search for words or phrases in your BPM library. The search functionality searches the names and descriptions of business processes.

- To search for a **word**, enter the search word in the search box, and then press Enter.
- To search for a **phrase**, put double quotation marks around the search phrase.
  
  For example, enter **technology** (word) or **"information technology"** (phrase) in the search box.

- You can also search for Application Object Tree (AOT) elements that are part of the task recordings that are in your library. Typically, these AOT elements are the names of pages or menu items. When you search for an AOT element, prefix it with a dollar sign ($). For example, enter **$CustTable** in the search box.
Upload a task recording

1. In Microsoft Dynamics Lifecycle Services (LCS), in your project, on the Business process libraries page, select the library to upload the task recording to.

   ![Business process libraries](image)

   - In Microsoft Dynamics Lifecycle Services (LCS), in your project, on the Business process libraries page, select the library to upload the task recording to.

   - Select the process to upload the task recording to.

   ![Overview](image)

   - On the Overview pane, select Upload. Select Browse to find and select the file to upload, and then select Upload.
Download a task recording

You can download a task recording (AXTR file) that has been uploaded to a BPM process.

1. In your LCS project, on the Business process libraries page, select the library to download the task recording.

2. Select a process that has task recording uploaded.

3. On the Overview pane, select Download to save the task recording (AXTR).

Export a methodology to Word

1. In your LCS project, on the Business process libraries page, select the library to export.

2. Select the process to export, and then, in the right pane, select Doc to begin the download.

   NOTE
   The methodology will begin from the process step that you selected.

Publish a BPM library

- In your LCS project, on the Business process libraries page, on the tile for the library that you want to copy, select the ellipsis button (...), and then select Publish.
Distribute a BPM library

When you distribute a BPM library, the library will be available to all users who are a part of your organization. In other words, it will be available to all users who sign in to LCS by using your organization's domain (for example, all users who have an @contoso.com account).

1. Ask the customer to invite you to their project.
2. Sign in to the customer's LCS project by using your organization's account.
3. On the **Business process libraries** page, copy the library from the **Corporate libraries** pane to the customer's project.
You can associate an activity diagram with a business process. Activity diagrams are used to describe how a business process or task is completed in a proposed software solution.

There are two types of activity diagrams:

- **Task recordings** – Business processes that are associated with task recordings for Finance and Operations, include activity diagrams and process steps that are automatically generated.

- **Microsoft Visio** – You can associate a business process with a Visio diagram by manually uploading a Visio file.

### Browse activity diagrams

The **Diagrams** column in your BPM library indicates whether a particular business process is associated with an activity diagram. The number in the column indicates the number of child processes that include diagrams. The symbol next to the number indicates whether the current node or process is associated with a diagram. These indicators don’t apply to Visio diagrams.

To view an activity diagram, select the business process, and then, in the right pane, on the **Overview** tab, select **Diagrams**. The **Flowchart** page appears.

### Upload Task Recording

To upload a task recording, open the business process library that you want to upload to. Select the process step that you want to upload the task recording to, and then click **Upload**.
Activity diagrams that are created from task recordings

[IMPORTANT] Flowchart diagrams in Business process modeler have been deprecated. To learn more about the deprecation, see Flowchart diagrams in Business process modeler.

You can create a task recording in your environment and save it directly to Microsoft Dynamics Lifecycle Services (LCS). In this way, you can associate the task recording with a business process in a BPM library. For more information, see Connecting the help system and Create documentation or training with Task Recorder.

The Task recorder tool lets you create a distributable recording file. Recording files have the .axtr file name extension. You can associate a business process in BPM with a task recording by manually uploading the recording file.
To upload a recording file, select the business process, and then, in the right pane, on the **Overview** tab, select **Upload**.

BPM automatically generates an activity diagram and detailed process steps for all task recordings that are created. The following illustration shows an example.

Visio files

You can associate a business process with a Visio diagram. Typically, this functionality is used for high-level processes that can't be represented by a task recording.

**NOTE**

BPM supports .vsd and .vsdx files. However, it doesn't support .vsdm files (macro-enabled Visio drawing files). If a .vsd file contain macros, BPM disables the execution of the macros.

To view or upload a Visio file, follow these steps.

1. Select the business process, and then, in the right pane, on the **Overview** tab, select **Diagrams**.
2. On the **Flowchart** page, select the **Visio** tab. For more information, see the "Unconnected flowcharts" section in [Flowcharts in Business process modeler (BPM)](link).
You start the implementation stage of a project by synchronizing a Business process modeler (BPM) library with your project in Microsoft Azure DevOps. In this way, you can review processes and associate requirements with business processes. By synchronizing a BPM library with an Azure DevOps project, you can also track the progress of your implementation project in Azure DevOps, and can associate various work items with requirements and business processes. These work items include bugs, tasks, backlog items, tests, and documents.

Currently, BPM-Azure DevOps synchronization doesn’t support custom work item types or synchronizing business processes with custom work item types. If you try either of these, you will receive a warning. If you choose to ignore the warning and attempt a Azure DevOps sync with a custom template, you can avoid synchronization issues by verifying the following for the template:

- Does not delete any work item type
- Does not delete any state of a work item type
- Does not add any required fields to a work item type

To learn more about Azure DevOps, go to [www.visualstudio.com/team-services](http://www.visualstudio.com/team-services).

**LCS project settings: Set up Azure DevOps**

If you’ve already set up Azure DevOps from Microsoft Dynamics Lifecycle Services (LCS), you can skip the procedures in this section.

**Create a personal access token**

To connect to a Azure DevOps project, LCS is authenticated by using a personal access token. Follow these steps to create a personal access token in Azure DevOps.

1. Go to [https://www.visualstudio.com](https://www.visualstudio.com), sign in, and find your Azure DevOps project.
2. In the upper-right corner, hold the pointer over your name, and then, on the menu that appears, select Security.
3. Select Add to create a new personal access token.
4. Enter a name for the token, and then specify how long the token should last.
5. Select Create Token.
6. Copy the token to your clipboard.

**NOTE**

You won’t be able to find the token details again after you complete this step and move away from the page. Therefore, make sure that you’ve copied the token before you move away from the page.

**Configure your LCS project to connect to Azure DevOps**

1. In your LCS project, select the Project settings tile.
2. Select Azure DevOps, and then select Setup Azure DevOps. This configuration is required by many LCS tools. If you’ve already configured LCS to connect to your Azure DevOps project, you can either skip
this procedure or select **Change** to change the existing configuration.

3. Enter the root URL for your Azure DevOps account, and the personal access token that you created earlier, and then select **Continue**.

4. Select your Azure DevOps project.

5. Specify the mapping between LCS/BPM items and the associated Azure DevOps work item types.

   **Setup Visual Studio Team Services**

   1. Enter the Visual Studio Team Services site URL to allow Lifecycle Services to connect and manage resources.

   2. Select the Visual Studio Team Services project to choose the Visual Studio Team Services project in the selected site to link with the Lifecycle Services project.

   3. Select the Azure DevOps project.

   4. Enter the root URL for your Azure DevOps account, and the personal access token that you created earlier, and then select **Continue**.

   5. Set the mapping between LCS/BPM items and the associated Azure DevOps work item types.

   6. Select **Continue**, review your changes, and then select **Save**.

**Synchronize a BPM library with an Azure DevOps project**

After you've set up the connection between the LCS project and an Azure DevOps project, you can synchronize a BPM library with an Azure DevOps project. When you synchronize a BPM library with an Azure DevOps project, a Azure DevOps work item is created for each business process line in the BPM library. In addition, the hierarchy of business processes in BPM is reflected in the hierarchy of work items in Azure DevOps. The type of work items that are created in Azure DevOps depends on the settings of your LCS project.

This synchronization is a one-way synchronization. Changes in LCS are reflected in Azure DevOps, but changes in Azure DevOps aren't reflected in LCS.

The following information is synchronized:

- Business process names
- Business process descriptions
- Keywords (as tags)
- Countries or regions (as tags)
- Industries (as tags)

To synchronize a BPM library with an Azure DevOps project, on the **Business process libraries** page, on the tile for the library that you want to synchronize, select the ellipsis button (…), and then select **Azure DevOps sync**.
You can also start Azure DevOps synchronization from the toolbar in a BPM library. Select the ellipsis button (…), and then select Azure DevOps sync.

NOTE
BPM localization is not supported. If you edit in the new BPM client in any language other than EN-US, your changes will only display when you view the BPM in the language in which the changes were made. To view any changes made in EN-US, you must synchronize with Visual Studio Team Server before the changes will display.

Turn off synchronization of BPM with Azure DevOps
To turn off synchronization, on the Business process libraries page, select the library that you want to stop synchronizing, select the ellipsis button (…), and then unselect Azure DevOps sync.

Review processes and add requirements
During the project phase where you’re gathering requirements, you can use the BPM library to review business processes and tasks, and to identify requirements. In BPM, you can mark business processes as reviewed to track the review process.

To mark a process or one of its child processes as reviewed, select the process in BPM, and then, in the right pane, on the Overview tab, select Mark as reviewed.

When a business process is marked as reviewed, the Reviewed column is updated. This column shows the following information:

- A fraction indicates how many direct child processes have been reviewed.
- A symbol indicates how completely the process and its child processes have been reviewed:
  - Green check mark – The process and all its child processes have been fully reviewed.
  - Yellow circle – The process and its child processes have been partially reviewed.
  - Red dash – The process and its child processes haven’t been reviewed.
While you're reviewing a business process that is connected to Azure DevOps, you can add a requirement directly to your Azure DevOps project.

1. Select a business process.
2. In the right pane, on the Requirements tab, select Add requirement.
3. Enter a name, description, and type, and then select Create.

   In Azure DevOps, a requirement work item is created that is associated with the current business process.

To go to the Azure DevOps work items that are associated with the current business process, on the Requirements tab, select the appropriate links.

**Common syncing errors**

If the BPM to Azure DevOps synchronization fails, you will see the failed process name, work item type, and an error message.

Here are some common causes and suggested actions to resolve the error.

<table>
<thead>
<tr>
<th>POSSIBLE CAUSE</th>
<th>ERROR MESSAGE</th>
<th>SUGGESTED SOLUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required field added</td>
<td>Failed to create work item. A required field has been added to this work item type, which is not supported. Remove this requirement or provide a default value in the process template to unblock the operation.</td>
<td>Remove the required field or provide a default value.</td>
</tr>
<tr>
<td>Work item type disabled</td>
<td>Failed to create work item. The work item type has been disabled in the process template. Enable the work item type to unblock the operation.</td>
<td>Enable the work item type in the process template</td>
</tr>
<tr>
<td>POSSIBLE CAUSE</td>
<td>ERROR MESSAGE</td>
<td>SUGGESTED SOLUTION</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Couldn't find work item to update</td>
<td>Failed to update work item. The work item does not exist, or you do not have permissions to read it. Check the PAT configuration in the project settings or restore the work item if it has been deleted directly from the DevOps project.</td>
<td>Restore the work item from the recycle bin if it was deleted, or create a new Personal Access Token (PAT) and make sure that it has full permissions.</td>
</tr>
<tr>
<td>Personal Access Token is expired</td>
<td>Failed to sync with Visual Studio Team Services. The request response is: Unauthorized. Please check that the PAT is setup correctly and still valid, try again and contact support if the error persists.</td>
<td>Create a new Personal Access Token (PAT) from Azure DevOps and update the PAT value in your LCS Project settings.</td>
</tr>
<tr>
<td>Generic error</td>
<td>Failed to sync with Visual Studio Team Services. The request response is: {0}. Please check that the PAT is setup correctly and still valid, try again and contact support if the error persists.</td>
<td>Contact customer support with the request response that caused the syncing error.</td>
</tr>
</tbody>
</table>
You can use Task recorder and Business process modeler (BPM) to create user acceptance test libraries. Task recorder is a powerful tool to record test cases and organize them by business process using BPM. As a Microsoft partner you can use BPM to distribute test libraries to your customers via LCS and LCS solutions. If you are a customer, use BPM to author and distribute test libraries across different projects and team.

Because BPM can be synchronized with Azure DevOps (formerly known as Visual Studio Team Services), you can automatically create test cases (including test steps) in your Azure DevOps project. Azure DevOps can then serve as your test configuration and test management tool where you can create targeted test plans and test suites, manage the execution of tests and investigate results. For more information about testing with Azure DevOps, see What are test plans, test suites, and test cases?

This topic walks through the process of creating and executing acceptance test suites to be used for manual or automated testing.

Create a Scenario Acceptance Testing BPM library

BPM is a great LCS tool to describe a hierarchy of business processes and user tasks. LCS also allows Microsoft partners and customers to author and distribute BPM libraries across LCS projects via the Asset library. This section describes how to take advantage of BPM to define your acceptance test library.

Create a BPM library

There are several ways to create a Business process modeler (BPM) library. For more information about how to create libraries in BPM, see Create, edit, and browse Business process modeler (BPM) libraries.

For illustration purposes, this topic uses a library that contains common business processes, such as create an expense report and approve order requests. The library was created by using the Excel import functionality.

Record test cases and save to BPM

After you have created a BPM library, you'll need to use Task recorder to create your test cases and then upload the cases to BPM. There are several ways to do this.

If you're using a BPM library that already has all of the necessary task recordings (test cases) attached, you can skip this step. Otherwise, follow the instructions below to create new task recordings.
Create and save a new task recording

1. Open the client and sign in.
2. Select the company that you want to use while recording.
3. Go to Settings > Task recorder.

4. Click Create a new recording.
5. Enter a name for the recording, and then click Start. Recording begins the moment that you click Start.
6. When the recording is complete, in the Task recorder pane, click Stop.
7. To save the task recording to an attached BPM, click Save to Lifecycle Services.
8. Select the library that you want to save the recording to, and then click Save. Otherwise, select Save to Disk and follow the steps in the next section, "Upload an AXTR file to BPM."

NOTE
To enable the effective execution of your tests using automation tools, make sure all of your task recordings start on the main dashboard of your application. For end-to-end processes that are performed by more than one user, we recommend that you divide your task recordings into user-specific tasks. This simplifies the maintenance of test cases and allows you to execute test cases in the context of security roles, which is a best a practice.

Upload an AXTR file to BPM
If you have saved your recordings (AXTR files) to disk, follow these steps to upload them to BPM.
1. In Lifecycle Services (LCS), in your project, on the Business process libraries page, select the library to upload the task recording to.
2. Click Author and edit and in the lines, locate and select the process to upload the task recording to.
Save an existing task recording to BPM

1. To attach an existing task recording, sign in to the client.
2. Go to Settings > Task recorder.
3. Select Edit Task Recording and attach the file by either saving directly to LCS or downloading the AXTR and then uploading to BPM.

Guidelines for recording test cases

Follow these guidelines when authoring and recording your test cases, especially if you are planning to automate test execution. The process and tools described in this article apply to business process acceptance tests. They are not meant to replace component and unit testing that is typically owned by developers.

- Author a limited number of test cases that, when combined, cover complete end-to-end processes.
- Focus on business processes that have been customized.
- An individual test case (recording) should cover one or two business tasks only, typically executed by one person. This simplifies task recording maintenance. Do not combine a complete end-to-end business process such as “Procure to Pay” or “Order to Cash” into one large task recording. For example, instead of having RFQ > Purchase Order > Product Receipt > Vendor Invoice > Vendor Payment as one test case, divide the process into three or four test cases. You will have the opportunity to combine these tests into an ordered
test suite later.

- A test case should have at least one validation. Try to validate critical fields that cover the impact of other fields. For example: Validation of totals on sales or purchase orders cover the unit price/quantity/discount/tax...etc.
- Avoid printing a report in a test case. If a test case needs to print a report, it should be selected on screen.
- 80+% of test cases should be of transactions or source documents. Master data should be limited to up to 20% of test cases only.

**Synchronize and configure your test plan in Azure DevOps**

An acceptance test library is your starting point. It typically contains all test cases (task recordings) of a particular application organized by business process. During a particular test pass, you usually do not need to execute all test cases. What test cases you select depends on the phase of your implementation or the nature of the update you are planning to apply to your production environment. Azure DevOps enables you to organize your test cases in test plans and test suites. A test plan contains one or more test suites (A subset of your test library); test cases can belong to more than one test suite.

Once you have selected your acceptance testing BPM library, synchronize it with Azure DevOps and create your test plan and test suites.

**Sync with Azure DevOps**

Synchronize your BPM library with your Azure DevOps project. For more information, see [Synchronize BPM libraries with Azure DevOps](#).

After configuration is complete, synchronize the BPM library with a Azure DevOps project.

1. On the Business process libraries page, on the tile for the library that you want to synchronize, select the ellipsis button (…), and then select **Azure DevOps sync**.

2. After Azure DevOps synchronization is complete, select the ellipsis button (…), and then select **Sync test cases**.
Create a test suite in Azure DevOps

3. When this step is complete, your task recordings will become test cases in Azure DevOps and a link will appear under the **Requirements** tab.

   ![DemoLibrary](image)

   In addition to the test steps, the task recording XML file is attached to the Azure DevOps test case. This file will be needed if you want to automate test execution.

   **Create a test suite in Azure DevOps**

   Next, you will need to create a test plan and test suite in Azure DevOps. This will allow you to execute an ordered suite of test cases and easily manage, investigate, and track the results.

   1. Sign in to Azure DevOps and select the project and test plan that you want to test in.

   2. On the toolbar, select **Test > Test Plans**.

   3. In the left pane, select `+`, and then select **Static suite**.

   4. Enter a name for the suite.

   5. Click **Add existing** and query the tag LCS:Test Cases.

   6. Click **Run > Add test cases**.
7. Select the test case to view details and the attached XML file.

NOTE

This example shows how to create one comprehensive acceptance test suite with all test cases added. Instead, you should create various test suites under the same test plan and then use custom queries to add specific test cases to a test suite. A test case can belong to more than one test suite.

Execute your tests

Run manual test cases

After you have a test suite, you are ready to use it for regression testing after updates have been made to your application in a sandbox or test environment. You can run the test cases in your test suite manually or play the task recordings that are part of the test suite and use Azure DevOps to mark the test cases as passed or failed.
Azure DevOps also provides a tool, **Test Runner**, to manage manual test case execution. For more information about using Test Runner, see [Run manual tests](#).

We recommend that you take advantage of Azure DevOps as it provides a rich set of management features not only for testing, but result management and mitigation.

### Run automated test cases

The platform for Finance and Operations provides developers with tools to author test cases based on task recordings and use Azure DevOps to manage the automated execution of these test cases.

Developers can use the build and test automation capabilities of **build and test** environments. For details, see the [Continuous delivery home page](#).

Functional power users can automate the execution of their test cases using the **Regression suite automation** tool. For more information, download the tool and read the [Regression suite automation tool](#).

### Investigate test runs

Once an automated run is complete, on the Azure DevOps toolbar, select **Test > Runs** (or **Test Plans > Runs**) to investigate your test run. Select the desired test run to investigate test case failures and errors. You can also go to your test suite in Azure DevOps to see the latest results associated with your test cases. For more information on testing and test management in Azure DevOps, see the [Azure DevOps documentation](#).
You can use Business process modeler in Microsoft Dynamics Lifecycle Services (LCS) to define and store business process flowcharts for an organization. This topic explains how you can view the default connected flowcharts, export a connected flowchart as a Visio file, and upload and view unconnected flowcharts.

- Connected flowcharts are the automatically generated flowcharts based on data recorded in Task recorder and uploaded to Business process modeler, this also includes the process steps from the task recording.
- Unconnected flowcharts are uploaded directly from Visio.

**View a connected flowchart**

Default connected flowcharts are available for many nodes in the industry-standard libraries. You can view a connected flowchart to determine whether it meets your needs.

To view a connected flowchart, follow these steps:

1. Sign in to Lifecycle Services, open a project, and then click **Business process modeler**.
2. In the **Project libraries** section, select a library to display it.
3. Expand the business process library and then click a library node that has a flowchart icon associated with it:

   ![Flowchart Icon]

   The flowchart is displayed. Each activity in the process is represented by a shape in the diagram. Process steps are displayed in the right pane.

**Export a flowchart as a Visio file**

You can export a business process model flowchart to a Visio file.

1. Sign in to Lifecycle Services, open a project, and then click **Business process modeler**.
2. In the **Project libraries** section, select a library to display it.
3. Expand the library and then select any library node that has a flowchart icon associated with it.
4. From the **Overview** pane, select **Diagram** to view the flowchart.
5. From the flowchart tab, click **Export** to save as a Visio file.

**Unconnected flowcharts**

Unconnected flowcharts, such as a Visio diagram, can be very helpful for describing high-level business processes that are performed outside of the Finance and Operations apps.

**Upload an unconnected flowchart**

1. In the **Project libraries** section, select a library to display it.
2. Expand the library and then click any library node that has a flowchart icon associated with it.
3. From the **Overview** pane, select **Diagram**.
4. From the Visio tab, click **Upload** to upload a Visio file.

**NOTE**

If you have uploaded the wrong file, you can delete the existing one, and upload a new one to replace it.
In Microsoft Dynamics Lifecycle Services, you can record information about custom business processes by using an updated version of Task recorder. You can then upload the files that you record to Business process modeler.

This topic explains where to find the updated version of Task recorder and how to upload the custom business process files that you record. The updated version of Task recorder is available as a hotfix. You can download the hotfix from the following sites:

- Microsoft Dynamics AX 2012 and Microsoft Dynamics AX 2012 Feature Pack – Knowledgebase article 2863182
- Microsoft Dynamics AX 2012 R2 – Knowledgebase article 2863182

For more information about how to work with the updated Task recorder, see [Task recorder update for Microsoft Dynamics AX 2012](#).

**Upload custom recorded business processes**

You can upload business process artifacts (*.axbpm files) to the business process library. These files are generated from Task recorder. After they are uploaded, you can view and modify the recorded processes in Business process modeler. To upload custom business processes that you recorded, follow these steps:

1. On the Project home page, click the Business process modeler tile.
2. On the Business process library page, click Upload in the My libraries section or the Corporate libraries section.
3. On the Upload page, select the industry and enter a name and description for the file that you are uploading. Click Upload, select the .axbpm file, and then click OK. The upload process can take some time. You can view the status of the upload on the Administration page.
4. After the business process file has been uploaded, you can view the business process framework from the Business process library page.

**Additional resources**

- [Business process modeler (BPM) in Lifecycle Services (LCS)](#)
- [Business process libraries in Business process modeler (BPM)](#)
- [Flowcharts in Business process modeler (BPM)](#)
For customers, partners, and Microsoft to be successful in this endeavor, we must ensure that most of the actions are self-serve with the Microsoft Dynamics Service Engineering (DSE) team managing by exception. To attain this self-serve mode, the Microsoft Product team continues to add more automation around the various features needed to operate an environment.

The Finance and Operations apps are managed services. This means that Microsoft is responsible for managing and operating the production environments. Microsoft’s Dynamics Service Engineering team is available 24 hours a day, 7 days a week, and 365 days a year to operate and manage our customers’ production systems.

Monitor and troubleshoot the health of your environment

A key tenant for a successful onboarding experience to the cloud service is knowing the health of your environments at all times and being able to troubleshoot health issues when necessary. Lifecycle Services (LCS), which is the admin center for Finance and Operations, contains a collection of monitoring and diagnostics tools which can help ensure that you have an accurate view of the environments that you manage. For more information, see Monitoring and diagnostics tools in Lifecycle Services (LCS).

Update your environment

After go-live, the Production environment must be updated at regular intervals. Lifecycle Services (LCS) provides a self-serve experience to continuously update your environments.

Update types

For customers who are on Dynamics 365 for Finance and Operations version 8.0 (April 2018) and earlier, the following updates are available:

- **Platform updates** – A single cumulative binary update of all the platform fixes.
- **Application hotfixes** – Application hotfixes that are released as granular X++ updates.
- **Application release** – A new major release of the application. This type of update typically requires an upgrade.
- **Application customizations** – Customizations that are built on top of the application. The best practice is to apply a single deployable package that consists of all your independent software vendor (ISV) solutions and customizations.

For customers who are on Dynamics 365 for Finance and Operations version 8.1 (October 2018) and later, the following updates are available:

- **Application updates** – A single cumulative binary update of the application and the platform fixes. You can update for yourself by using the regular update flows. Otherwise, you will be automatically updated by Microsoft.
- **Application customizations** – Customizations that are built on top of the application. The best practice is to apply a single deployable package that consists of all your ISV solutions and customizations.

Cloud infrastructure

Microsoft is responsible for managing the infrastructure for your environments. Therefore, some updates, such as operating system updates, must be done on a monthly basis in a planned maintenance window. Other kinds of updates might include changes to the infrastructure components.
Update policy

Currently, service updates require production tenant downtime and are applied in two kinds of maintenance windows.

- **Microsoft planned maintenance window** - Microsoft will provide customers with no less than five business days’ notice regarding upcoming planned maintenance downtime. The default downtime window is defined per region and is scheduled to occur over a weekend to minimize impact to the business. Cloud infrastructure updates can be done in this window. For more information about planned maintenance, see the Planned maintenance window FAQ.

- **Customer initiated maintenance window** - A customer selects the maintenance window through LCS as a part of the package application flow. Updates are done in this maintenance window.

**Search for and apply an update in Lifecycle Services**

Updates are applied as deployable package on an environment. A deployable package is a format that is used to apply updates to all the environments in a project. When you encounter an issue in the production environment, you can quickly find and apply a hotfix on all of the environments (Dev/Sandbox and Prod).

- **Search for and download an update** In LCS, you can search for an update using Issue search in Lifecycle Services (LCS) or the Download updates from Lifecycle Services (LCS). Because the steps to prepare an update differ based on the update type, after the update is downloaded, use the following list to determine how to proceed with preparation.
  - Platform update: Platform updates are cumulative and binary. This means that they can be applied directly to an environment. After the update is downloaded, it can be automatically applied to an environment by uploading it to the Asset Library.
  - Application hotfixes: Application hotfixes are code changes. After the application hotfix is downloaded, it must be applied on a dev environment to generate a deployable package. For more information, see Create deployable packages of models and Install metadata hotfixes in development environments.
  - Application customizations: These are customizations that ISV or partners create. These are deployable packages that are uploaded to the Asset Library and can be applied from there.

- **Apply an update** Use the information in the topic, Apply updates to cloud environments, to walk through the steps for applying a deployable package. The update package can be a binary hotfix for Application Object Server (AOS) or a deployable package that was created in your development environment.

- **Validate an update** After an update is applied, you should validate the application to:
  - Ensure that the update addressed the issue that it was applied for.
  - Verify that no regressions occurred from applying the update.
  - Verify that the build information was updated to reflect an update to the binaries.
    - For Platform updates, verify that the version of the AOS Service Model under Microsoft is updated.
    - For Application updates, check the version of the model that included the fix. For example, if the fix was in Application suite, then the version of the Application suite is updated.

**Upgrade your environment**

For information about how to upgrade to the latest version, see Process for moving to the latest update of Finance and Operations and What’s new or changed in Finance and Operations home page.

**Environment data management**

These are the options for managing databases, including the ability to copy a database from one environment to another or restore a database to a previous state. For more information, see Database movement operations home page.
Sign up for cloud operations notifications

When the status of the package application is changed, LCS sends a notification to all of the users in a project. Any additional stakeholders who should be notified must be specified in the notification list.

1. To add additional stakeholders, in LCS, in the Environment details view, click Notification list.
2. Add the email address of each user who must be notified, and then click Save.
A service request is a ticket that you use to request that the Dynamics Service Engineering (DSE) team perform a predefined set of tasks on your environments.

**NOTE**

Service requests are only required for environments that are Microsoft-managed. Most environments are self-service. For more information about environment types, see [Cloud deployment overview](#).

Do not use service requests for product issues. If you encounter a situation that doesn't fit into any of the tasks that are described in this topic, submit a support ticket instead. For more information about support tickets, see [Get support for Finance and Operations apps or Lifecycle Services (LCS)](#).

You can use Microsoft Dynamics Lifecycle Services (LCS) to submit service requests directly to the DSE team. You can also view which requests have been submitted, executed, and canceled for your environments.

**View service requests**

There are two ways to view service requests:

- **On the project dashboard, in the Environments section, select Service requests.**

![SA Lab - Implementation Project](image)

- Select the **Menu** button and select **Work items**. On the **Work items** page select the **Service requests** tab.
By default, the **Service requests** tab on the **Work items** page lists all requests that are currently active and requests that have been denied. However, you can use the filter options to show canceled and finished requests too.

After you submit a request, it has a status of **Requested**. Before the DSE team acts on the request, it might ask for clarification by entering a comment in the **Comment** field. For example, you might receive a comment from the DSE team if you request deployment of a production environment, but the data center differs from the data center where your sandbox environments are deployed. Carefully review the comments, and provide any required clarification in your own comment. To view the details of a specific request, or to submit comments for a service request, select the request ID.

If you signed up for LCS notifications, you receive an email when the status of a service request changes or a comment is entered.

If you submit a service request to the DSE team, and the action is outside the team’s scope, the service request will be denied. In this case, the reason for the denial and suggestions for further action are provided. For some typical examples of service requests that the DSE team will deny, see the “Denied service requests” section later in this topic.

### Create service requests

There are two ways to create a service request: automatically and on demand.

- **Automatically** – A service request is automatically created when you request deployment of an environment, or an application of a package.
- **On demand** – A service request is manually created when you enter a request for a database point-in-time restore, and some other services.

#### Automatically create a service request

- **Environment deployment** – To set up deployment options and submit a request to the DSE team to deploy a new environment, in the **Environments** section, select **Configure**.

- **Package application** – To apply a package to the production environment, on the **Environment details** page, select **Maintain**, select the package to apply, and then select **Schedule**. For more information, see Apply updates to cloud environments.
IMPORTANT
If your scheduled time overlaps with a planned maintenance window, you will receive the following warning message.

⚠️ There is a high likelihood that maintenance activity may be scheduled for this environment during this time. Overlapping environment operations with maintenance activity will cause issues and possibly cause extended downtime. Would you like to proceed with this operation?

Show diagnostic information

If you choose to continue deploying the package, the package deployment operation will be rolled-back in the event of conflict, as planned maintenance takes priority.

This restriction is applicable to Microsoft-managed IAAS environments only.

Create a service request on demand
Service requests that are created on demand aren’t explicitly accepted by the DSE team. They will be addressed during the specified downtime window unless the DSE team has entered a comment in the request or has had to deny the request. For details, review the comments in the service request.

Microsoft frequently reviews all incoming service requests. By selecting the correct type of service request for your scenario, you help the DSE team handle the request in a timely manner.

1. On the Work items page, on the Service requests tab, select Add.
2. In the Create request dialog box, select the type of service request to create. The options on the page then reflect the specific type of request that you selected.

- **Sandbox point-in-time restore request** – Select this request type to restore a non-production database to a specific point in time. For more information, see Database movement operations home page.

  **NOTE**
  If you need to restore a production database to a previous point-in-time during the cutover phase, select the Production point-in-time restore request type. If you need to restore a production database when you’re already live in operations, submit a support ticket through LCS.

- **Database refresh request** – Select this request type to refresh a database from a production environment to a sandbox environment, or from one sandbox environment to another. For more information, see Refresh database. This request type is being retired on January 31, 2019.

  **NOTE**
  If you need to refresh a database from a sandbox environment to a production environment during the cutover phase, select the Sandbox to Production type.

- **Sandbox to Production** – Perform a database refresh of your configuration data to a production environment.
Commonly denied service requests

- **Other request** – You need to use the **Other request** type exactly as described here. If you word a request in a way that isn’t clear to the DSE team, the team will enter a comment to ask for clarification, and your request will be delayed. If you use the **Other request** type for any request that isn’t listed below, the request will be denied. Select this request type to request that the DSE team perform one of the following actions:
  - Turn on maintenance mode in a production environment. For more information, see Maintenance mode.
  - Tenant move of a live Production environment. Request the Microsoft Service Engineering team to move the Production database and Azure Blob Storage from the old tenant to the new tenant if you are moving tenant on a live Production environment. Make sure that you only request this service when you are ready with all prerequisites. For more details, see Move LCS implementation projects to different Azure AD tenants.
  - Define explicit Internet Protocol (IP) safe list rules in a production environment.

  **NOTE**
  Support for explicit safe list rules is deprecated for self-service environments. For more information, see Removed or deprecated platform features.

  - Request that Microsoft Power BI Embedded be activated in a sandbox environment, Standard Acceptance Test environment, or production environment if you receive the following message: “Power BI embedded isn’t enabled. Please contact your system administrator.”

**Commonly denied service requests**

Here are some typical examples of service requests that will be denied:

- You submit a request of the **Other request** type for one of the following actions, but you should have submitted a support ticket instead:
  - You want to activate a new subscription estimate after you’re live in production or after you’ve requested a production environment.
  - You want to reset the Financial reporting data mart in a release that is earlier than Microsoft Dynamics 365 for Finance and Operations Financial reporting release 7.2.6.0.
  - You want to restore a production database after go-live.
  - You encountered an issue after the DSE team did an application upgrade.
- You submit a request of the **Other request** type for an action that you should have requested through a different request type. Examples include a database refresh in a non-production environment.
- You submit a request of the **Other request** type for an action that you should perform yourself. Examples include a database upgrade in a development environment.

**Service request types and SLAs**

<table>
<thead>
<tr>
<th>SERVICE REQUEST TYPE</th>
<th>APPLICABLE ENVIRONMENTS</th>
<th>REQUESTED SERVICE</th>
<th>LEAD TIME</th>
<th>DOWNTIME</th>
</tr>
</thead>
</table>

[Insert table data here]
<table>
<thead>
<tr>
<th>SERVICE REQUEST TYPE</th>
<th>APPLICABLE ENVIRONMENTS</th>
<th>REQUESTED SERVICE</th>
<th>LEAD TIME</th>
<th>DOWNTIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment deployment</td>
<td>Any</td>
<td>Environment deployment</td>
<td>Service level agreement (SLA): within two business days</td>
<td></td>
</tr>
<tr>
<td>Package application</td>
<td>Production</td>
<td>Deployable package application</td>
<td>Five hours</td>
<td>Five hours</td>
</tr>
<tr>
<td>Sandbox point-in-time restore</td>
<td>Any Tier 2 or higher sandbox</td>
<td>Database point-in-time restore</td>
<td>Five hours</td>
<td>Four hours</td>
</tr>
<tr>
<td>Production point-in-time restore</td>
<td>Production</td>
<td>Database point-in-time restore</td>
<td>Based on data volume</td>
<td>Based on data volume</td>
</tr>
<tr>
<td>Sandbox to Production</td>
<td>Tier 2 or higher sandbox to Production</td>
<td>Sandbox to Production</td>
<td>Five hours</td>
<td>Four hours</td>
</tr>
<tr>
<td>Other</td>
<td>Production</td>
<td>Maintenance mode</td>
<td>Five hours</td>
<td>Not applicable, because the customer indicates in the service request when the environment should be taken out of maintenance mode again</td>
</tr>
<tr>
<td></td>
<td>Production</td>
<td>IP safe list rules</td>
<td>Five hours</td>
<td>Two hours</td>
</tr>
<tr>
<td></td>
<td>Production</td>
<td>Power BI Embedded</td>
<td>Five hours</td>
<td>Two hours</td>
</tr>
</tbody>
</table>
What is a planned maintenance window?
A planned maintenance window is the timeframe that Microsoft has scheduled to apply infrastructure or service updates to your cloud service.

How does a planned maintenance window work?
For planned maintenance scheduled on your Tier 2 through Tier 5 sandbox environments and production environments, Microsoft will send a notification to all stakeholders five business days before the start of the patching window. The patching window is the period when the environment is patched. It’s defined by geographic region. Details about the maintenance activity will be included in the notification that is sent to stakeholders. For Microsoft-managed Tier 1 environments, we will not send any notifications before the update.

When is this planned maintenance window taken?
To limit the impact on users, the maintenance window is planned according to the region where environments are deployed. The following list shows the maintenance window for each region. All environments fall into one of these three regions. The times are shown in Coordinated Universal Time (UTC, which is also known as Greenwich Mean Time).

- **NAM:** 2 AM to 10 AM
- **EMEA:** 10 PM to 6 AM
- **APAC:** 12 PM to 9 PM

Will the maintenance from Microsoft require any uptake?
Most of the maintenance operations require no action on your end. If there is a critical security update that requires uptake, you will be notified.

Who will be notified about the upcoming planned maintenance?
The following stakeholders will be notified about the upcoming maintenance:

- Project owners
- Organization admins
- Environment admins
- Other people who are specified in the list during deployment or through the Notify button on the environment details page in Microsoft Dynamics Lifecycle Services (LCS)

How do I sign up to be notified about the maintenance window?
Any partner, independent software vendor (ISV), and other interested party who wants to be notified about upcoming updates can request to be added to the LCS project as a relevant stakeholder (project owner, environment admin, or additional stakeholder).

Why can't these updates be applied in zero downtime?
Microsoft is continually working to reduce the necessity of downtime for the service, and many regular maintenance tasks don't incur downtime. However, to help guarantee the most predictability, Microsoft can't yet do all patching in zero downtime.

Microsoft service updates
A separate set of frequently asked questions (FAQ) provides details about service updates that are done by
Infrastructure updates

How long is the maintenance window?
Most operating system–level updates are completed in approximately one hour. However, Microsoft asks for a three-hour window, so that there is time to handle any failures and to bring the system back to a healthy state.

The exact downtime for all updates will be included in the maintenance window notification email that is sent to you before the start of the update.

How frequent are the updates?
Operating system–level updates are applied monthly to your Microsoft-managed Tier 2 through Tier 5 sandbox environments and to the production environments. However, Microsoft-managed Tier 1 environments are updated weekly in the maintenance windows defined per region.

Where can I learn more about what is applied?
For more information about the updates that will be applied, see Microsoft Security Bulletins.

Where can I track progress of the update?
During operating system–level updates, LCS doesn't currently indicate that any patching is in progress. However, Microsoft plans to add this functionality at some point.

What environments are updated?
Operating system–level updates are applied to all Microsoft-managed environments that are included as part of the Microsoft base offer. This includes your Tier 1, Tier 2 through Tier 5, and Production environments. They are also applied to add-ons that have been purchased. However, other environments, such as environments hosted in your subscription (known as Cloud hosted environments), are the responsibility of the customer or partner.

What notifications will I receive about upcoming planned maintenance?
You will receive a notification email for the scheduled update on your Tier 2 to Tier 5 sandbox environment and production environment, five days before the update is scheduled to occur. This email will include information about the environments that will be updated, the update type, the estimated amount of time that the update will take, and any action that you might have to take. We do not send notifications for Microsoft-managed Tier 1 environments.

Will I be notified when the update is completed?
If your update is completed within the defined maintenance window, you won't receive any notification when the update is completed.

How do I report an issue that is identified during validation of the updates that were applied to the environment?
To report an issue that is identified during update validation, file a support ticket with Microsoft and append the title with 'Planned Maintenance Window'.

What happens if the patching fails?
If the patching fails during an operating system–level update, the specific patch is skipped and will be applied in the next update cycle.

Will I be compensated if the update takes longer than the scheduled maintenance window?
If the update takes longer than the scheduled maintenance window, the extra time is considered unplanned downtime and is subject to the general service level agreement (SLA).

How do I reschedule security maintenance activities?
Security maintenance helps ensure a secure environment, and not doing the maintenance could potentially
introduce an avoidable security risk.

If there is an absolute business need and you are unable to move forward with this maintenance during the timeframe listed above, you can request to reschedule the current maintenance activity for Tier 2 through Tier 5 sandbox environments and production environments by filing a support ticket with Microsoft. The deadline for filing the support ticket to request a reschedule will be included in your notification email. Any request submitted after that deadline will not be honored.

We do not offer rescheduling of security maintenance activities on Microsoft-managed Tier 1 environments.
This topic describes the various tools that Microsoft Dynamics Lifecycle Services (LCS) provides to help you monitor, diagnose, and analyze the health of the Finance and Operations environments that you manage.

To have a successful onboarding experience to the cloud service, you must know the health of your environments at all times. You must also be able to troubleshoot any health issues that occur. Microsoft Dynamics Lifecycle Services (LCS), which is the administration center, contains a collection of monitoring and diagnostics tools that can help to ensure that you have an accurate view of the environments that you manage.

**Telemetry data**

The telemetry data that is the basis of the Monitoring and diagnostics portal in LCS has three primary use cases: monitoring, diagnostics, and analytics.

- **Monitoring**
  - High scale, low latency, reliable alerts
  - Hot path < 60 seconds

- **Diagnostics**
  - Interactive visualization with Near-Real-Time (NRT)
  - Warm path < 5 minutes

- **Analytics**
  - App insights, data analytics and reports
  - Cold path > 5 minutes

**Monitoring**

In business operations software, you should always know whether your environment is up and running, so that it can perform business operations. You should also be able to easily view the health of the environment through LCS. Microsoft supports two types of monitoring capabilities:

- **Availability monitoring** – This type of monitoring performs a check against the environment to make sure that it’s available at all times. If the check fails, the Microsoft Service Engineering team is immediately notified.

- **Health monitoring** – In addition to availability checks, some basic health checks must be performed. These health checks span various components, such as Application Object Server (AOS), Batch Framework, Data Management Framework, Microsoft Azure SQL, and Management Reporter. These checks are done based on multiple data sources, such as the telemetry that is collected from the environments, checks that are done by a watchdog service that continuously monitors the environment, and CPU counters and other system-level counters that the environment emits. Some health checks are self-healing and are mitigated immediately. However, other health checks are reported to the Microsoft Service Engineering team for investigation.

**Diagnostics**

When a user reports an issue, you can use various tools in LCS for troubleshooting. The rich set of telemetry data helps you build a storyboard view that shows what that user and other users were doing when the issue was reported. In addition to user activity tracking, a rich set of SQL data is available for performance troubleshooting.

**Analytics**

Analytics is another critical use case for the telemetry data that is collected. Currently, only Microsoft can perform analytics, so that it can gauge and understand feature usage and performance through Microsoft Power BI.
Responsibilities

For a managed cloud service such as Finance and Operations, Microsoft is responsible for actively monitoring the health of production environments at all times. If a customer’s environment is affected by an issue, the Microsoft Service Engineering team is immediately alerted. The team will start to investigate the issue and will work with you to find a resolution. However, you’re responsible for proactively or reactively monitoring and troubleshooting the health of non-production environments.

Access the Monitoring and diagnostics portal

1. Open LCS, and navigate to the appropriate project.
2. In the Environments section, select the environment to view, and then select Full details.
3. On the environment details page, select Environment monitoring to open the Monitoring and diagnostics portal.

Tools

Several tools and resources are available in the Monitoring and diagnostics portal.

**NOTE**

Not all environments contain all the tools. The following table shows the tools that are available for each type of environment.

<table>
<thead>
<tr>
<th>ENVIRONMENT TYPE</th>
<th>TOOLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Production systems</td>
<td>• Activity monitoring</td>
</tr>
<tr>
<td></td>
<td>• Environment monitoring</td>
</tr>
<tr>
<td></td>
<td>• SQL insights</td>
</tr>
<tr>
<td></td>
<td>• System diagnostics</td>
</tr>
<tr>
<td>User acceptance testing (UAT)/sandbox</td>
<td>• Activity monitoring</td>
</tr>
<tr>
<td></td>
<td>• SQL insights</td>
</tr>
<tr>
<td></td>
<td>• System diagnostics</td>
</tr>
</tbody>
</table>
Environments deployed in customer/partner subscriptions

<table>
<thead>
<tr>
<th>ENVIRONMENT TYPE</th>
<th>TOOLS</th>
</tr>
</thead>
</table>
| Demo/build                            | - Activity monitoring  
|                                       | - System diagnostics  |
| Environments deployed in customer/partner subscriptions | - System diagnostics |

**Monitoring dashboard**

On the **Environment monitoring** page, select the **Health metrics** tab to view the **Monitoring** dashboard. Health metrics are collected for every machine and component. These health metrics include CPU usage, available memory, errors logged per second, and batch heartbeat. You’re alerted about any abnormalities in the metrics. Although some alerts are self-healing, the Microsoft Service Engineering team will investigate the cause of other alerts and then take action to mitigate them. You can view the health monitors for a specific area to see what is occurring.

**Activity monitoring**

On the **Environment monitoring** page, select the **Activity** tab to use the Activity monitoring tool. This tool provides a storyboard view that shows what you or another user was doing during a specific period.

![Environment monitoring](image)

- The **User interaction** chart shows a user’s activities on various machines in the environment and the SQL utilization trend.
- The **User load** section shows all the system users. Each chart shows the time that the user spent on a specific machine.
- The **Activity load** section shows the activities that were performed on each machine. If you hover over an activity, you see the Form:Control:Action as a tuple. For example, if you look at LedgerJournal:New:Click in this section, you can see that user A opened the LedgerJournals page and selected the New button to create a new journal entry.
- The **User activity** grid shows the various activities that users performed, based on their session timestamp.

You can use the filters on this page to narrow the information logs. Here are some of the filters that are available:

- **Time duration** – Go back 60 minutes from the selected date and time.
- **User** – View a specific user’s activities.
- **Search terms** – Create a search that is based on the issue that is being investigated.
NOTE
The page doesn't load data by default. To load the data that is required in order to show the page, you must select the time duration and then select Submit time.

IMPORTANT
The Activity monitoring tool retains data for only 30 days.

Raw information logs
For advanced troubleshooting, you can view raw information logs. You can use a set of predefined queries to get raw logs for an issue. You can then export the logs to do more advanced analysis. The following types of queries are available:

- Slow queries
- Deadlocks
- Crashes
- Financial reporting issues

For information about how to use Azure Data Explorer with raw information logs, see Use Azure Data Explorer to query raw information logs.

SQL insights
The Monitoring and diagnostics portal also includes advanced SQL troubleshooting tools to enable performance analysis. Some of these tools are similar to the DynPerf tool that was used for SQL troubleshooting in Microsoft Dynamics AX 2012. For more details, see Performance troubleshooting using tools in Lifecycle Services (LCS).
Use Azure Data Explorer to query raw information logs

There are occasions when a customer, partner, consultant, or support engineer needs to look at the low-level telemetry data for a Finance and Operations app. These use cases include troubleshooting of errors, performance-related investigations, or just trying to gain some additional understanding of how the Finance and Operations app works. Telemetry data can be accessed by authorized users via the Environment monitoring features of Lifecycle Services (LCS) and can be filtered in a few different ways and displayed inside the LCS’s raw information logs. A data grid can be used to inspect the log entries. LCS does not allow for more sophisticated pivoting, so users can use Excel for that purpose. The telemetry data can also be downloaded and formatted in CSV format.

Excel is not the optimal tool for advanced querying of this data. A tool that is better designed for this purpose is the Azure Data Explorer. It provides an innovative query language, Kusto, that is optimized for high-performance data analytics. Answering questions like how often a certain process has occurred, how long has it taken in 90% of the occurrences, how often per hour has a certain action taken place over the course of a day becomes a lot easier and can be backed up with powerful graphics as well.

Here are two examples of how these types of graphics could look.

A lesser-known feature of the Azure Data Explorer is that it supports CSV files. You can use Azure Data Explorer to get CSV data files uploaded and staged so they can be queried with the Kusto language. To set up Azure Data Explorer cluster, see Quickstart: Create an Azure Data Explorer cluster and database.

Steps to upload to Azure Data Explorer

To upload to Azure Data Explorer, follow these steps:

1. Run your query on the LCS raw logs page.
2. Export the grid to Excel.

3. Open the file in Excel and save it without making any changes (this will fix any formatting issues).

4. In Azure Data Explorer, right-click the cluster in the tree view and select Ingest new data. On the next page, select Ingest data from a local file.

5. Select a cluster. Provide a name for the new table for the data to be imported into, and then select up to 10 CSV files to import. Select CSV format. Select Next until your data is imported.

6. Select the Query tile to display an area where you can write a Kusto query against your data.
Sample queries

**Analysis of SQL queries occurring in the Commerce Runtime (custom or built-in)**

```
new2
| where EventId == 1809 // designates SQL finished trace
| project executionTimeMilliseconds, sqlQuery
| summarize NumAllCalls=count(), TotalDuration=sum(executionTimeMilliseconds), AvgDuration =
  avg(executionTimeMilliseconds),
  percentiles(executionTimeMilliseconds, 90) by tostring(substring(sqlQuery, 0, 70))
| where percentile_executionTimeMilliseconds_90 > 1
| order by TotalDuration desc
```

**Performance of any RetailServer calls > 100ms**

```
// run this for the data
new2
| where EventId == 5009 // designates a RetailServer finished trace
| project executionTimeMilliseconds, apiAction
| summarize NumAllCalls=count(), TotalDuration=sum(executionTimeMilliseconds), AvgDuration =
  avg(executionTimeMilliseconds), percentiles(executionTimeMilliseconds, 90) by tostring(apiAction)
| where percentile_executionTimeMilliseconds_90 > 100
| order by percentile_executionTimeMilliseconds_90 desc

// include this for the chart
| project apiAction, percentile_executionTimeMilliseconds_90
| render columnchart
```

To learn more about the Kusto query language, see [Tutorial: Use Kusto queries in Azure Data Explorer and Azure Monitor](#).
You can use the Restart services functionality in Microsoft Dynamics Lifecycle Services (LCS) to restart individual services that are associated with a Tier 2, Tier 3, Tier 4, or Tier 5 standard acceptance test (sandbox) environment of the **Self-service** type. You can use this functionality to restart the following services:

- AX (the whole runtime)
- DIXF (the Data import export framework service)
- MR (the Financial reporting service)

Any user who has been added as a project owner, organization admin, or environment manager in an LCS project has permissions to use this functionality.

**Restart a specific service**

To restart a specific service in a deployed environment, follow these steps.

1. In LCS, open the appropriate project, and select the environment to restart the service for.
2. On the **Environment details** page, select **Maintain > Restart services**.
3. In the **Restart a service** dialog box, select the service to restart, and then select **OK**.
   
   The **Environment state** value is updated when the service is restarted.

4. To view the updated status, refresh the page.

**NOTE**

Because restart of a service might require only a few seconds, the **Environment state** value might already have been reset to **Deployed**. When the restart is completed, an entry is added to the **History** page.
Lifecycle Services (LCS) has a feature called Report production outage. This feature is available to all customers who have purchased one or more Dynamics 365 Finance and Operations apps and have implementation projects with a production environment deployed in LCS. This feature provides a quick and effective channel to escalate issues to Microsoft Support in the event that the services in a production environment are degraded or become unavailable.

Following mutually inclusive conditions, a production outage can be defined as one or more system-wide issues on a live production environment that impact multiple users and prevent your business from performing daily operations.

**Reporting flow**

The following list shows the order in which an issue should be handled:

1. In a live production environment, a customer experiences an outage or other situation with prevents business from continuing.
2. The customer reports a production outage issue by using the LCS Support portal.
3. The customer selects a production outage issue and provides additional information.
4. A Microsoft support engineer acknowledges the production outage ticket within 30 minutes of submission and begins to immediately collaborate with stakeholders to investigate and resolve the issue.
5. A support engineer contacts the customer to provide a status update.

**Access and availability**

All users who have been added to a customer’s implementation project have access to this feature. This includes project owners, organization admins, team members, and environment managers.

This feature is available for:

- Dynamics 365 Finance
- Dynamics 365 Supply Chain Management
- Environments that are managed by Microsoft
- A production environment in the LCS project
- All support plans

**Report a production outage**

To report a production outage, follow these steps:

1. Log in to your LCS project.
2. From the hamburger menu, click Support.
3. On the **Submitted To Microsoft** tab, click **Report production outage**.

4. Confirm the production outage, select the outage scenario from the drop-down list, and then click **Continue**.

5. Add a title and details about the outage, and then click **Next**.

6. Provide contact information, and then click **Next**.

7. Click **Done**.

If you're unable to report a production outage in LCS, **phone support** is available.

**NOTE**

If you don't see your situation listed in the outage scenarios, enter a support incident through LCS. During the initial investigation by a Microsoft support engineer, if it is found that the situation does not meet the current list of production outage scenarios, the support incident will be transferred to the correct support team and service-level agreement (SLA) based on your current support plan.
Many organizations are required to maintain an audit trail of users who have used the system. This requirement can be in place for compliance reasons, or to enable trackbacks in the event of incorrect use.

In Microsoft Dynamics AX 2012, the Audit log form recorded which users accessed the Microsoft Dynamics AX environment. In Microsoft Dynamics 365 Finance and Operations apps, this information is captured in telemetry. IT administrators can download this information by using Microsoft Dynamics Lifecycle Services (LCS) and then move it to offline storage to maintain the audit trail of users who have signed in.

To generate an audit log of users who have used the system, follow these steps.

1. Sign in to LCS, and open the project that is associated with your implementation.
2. Navigate to the production environment, and open the Environment details page.
3. On the Monitoring tab, select the Environment monitoring link to open the monitoring dashboard.
4. On the Activity tab, select View raw logs.
5. In the Query field, select User Login Events. You see a time duration that has a start date that is set to End date - 7 days.
6. Set the end date, and then select Search. The search results that are returned include all users who signed in to the system during the seven days before the selected end date.
7. The search results show the AADUserID value and the sign-in start and end times of the user’s session. To map the AADUserID value to the user’s user name and email address, use the Users page (System administration > Users).
8. To export the records and keep them for a longer period, select Export grid.

To help guarantee a complete audit trail, an IT administrator must complete this procedure every seven days.
The Asset library is a storage location for the various assets that are associated with a tenant in Microsoft Dynamics Lifecycle Services (LCS). Two types of Asset library are available in LCS: the Shared asset library and the project-level Asset library.

- **Shared asset library** – The Shared asset library is used by Microsoft and Partners to share assets across multiple tenants, projects, and environments in LCS. This library can be accessed by any user who signs in to LCS. To access the Shared asset library, sign in to LCS, and then click the Shared asset library tile.

- **Project-level Asset library** – The project-level Asset library is used to share assets across environments within a project in LCS. This library can be accessed by all users within a project. To access the project-level Asset library, sign in to LCS, and open a project. Then, on the hamburger menu, click Asset library.

**NOTE**

Uploading versions for the same asset in the project asset library is not supported.
The Asset library supports multiple types of assets. Here are some asset types that are frequently used:

- **Software deployable package** – This asset type represents all the packages that are used to update an environment with the latest set of updates.
- **Data package** – This asset type stores assets that are used for configuration and data management.
- **GER Configuration** – This asset type stores all localization and translation assets that are applied to the client.
- **Retail SDK** – This asset type stores all the latest scripts for the Retail software development kit (SDK).
- **Database backups** - This asset type is used for import and export of databases from Sandbox Tiers 2-5 environments.

**Asset scopes**

Every asset that the Asset library supports has multiple scopes. Here are some of the supported asset scopes:

- **Me** – When an asset is uploaded, it's set to the Me scope. An asset that has the Me scope is visible only to the person who uploaded the asset.
- **Project** – When an asset is imported from the Global scope to another project, it's set to the Project scope.
- **Organization** – When an asset must be shared with multiple users within a tenant, the tenant admin can promote the asset to the Organization scope.
- **Global** – Only Microsoft can upload assets to the Global scope. These assets are assets that Microsoft wants to be made publicly available to all LCS projects and users.

**Asset status**

Every asset has one of two statuses: **Draft** or **Published**.

- **Draft** – The asset can still be edited.
- **Published** – The asset is published at an Organization or Global scope, and edits are completed.

**Actions in the Asset library**

You can perform various actions in the Asset library as you require.

**Upload an asset to the Asset library**

1. Select the tab to upload the asset to.
2. Click the plus sign (+).
3. Enter a name and description for the asset.
4. Upload the file for the asset, and then click Confirm.

**Upload a new version for a specific asset (Shared asset library only)**

1. Select the asset in the Asset library.
2. On the toolbar, click the Upload new version button.
3. Repeat the steps in the previous procedure, "Upload an asset to the asset library."
4. On the toolbar, click Versions to view multiple versions for a single asset.
5. Individual versions can then be imported into a specific project asset library as required.

**Move assets from the Shared asset library to the project-level Asset library**

There are two ways to move an asset from the Shared asset library to the project-level asset library: you can import the asset or copy it.

**Import from the Shared asset library**

Follow these steps to import an asset from the Shared asset library to the project-level Asset library so that it can be applied across environments.

1. In the project-level Asset library, select the tab for the asset type to import.
2. Click **Import**.
3. In the list of assets in the Shared asset library, select the asset to import, and then click **Pick**.

The selected asset is imported and put into the project-level Asset library. The status of the asset in the project-level Asset library is set to **Published**. This method is for packages that you don't plan to edit. If you want to edit an imported package, create a copy by using the following procedure. The status of the package will then be **Draft**.

**Copy from the Shared asset library**
Follow these steps to create a copy of an asset so that it can be edited.

1. In the project-level Asset library, select the tab for the asset type to copy.
2. Select the asset to copy, and then, on the toolbar, click **Copy**.

A copy of the published asset is created, and the status is set to **Draft**.

**Save to my library**
After you've edited an asset, follow these steps to move the edited asset back to the Shared asset library so that it can be promoted to the **Organization** scope and shared with multiple customers.

1. In the project-level Asset library, select the tab for the asset type to import.
2. Select the asset to save, and then click **Save to my library**.

The asset is saved from the project-level Asset library back to the Shared asset library, and the scope is set to **Me**.
This topic describes how you can troubleshoot and mitigate performance issues using the tools available in Microsoft Dynamics Lifecycle Services (LCS).

Overview

Common feedback from customers and partners has been that they are unable to successfully diagnose performance issues using the tools in LCS. We have addressed this feedback by creating a more reliable way to collect performance metrics on demand. This enables customers and partners to execute a predefined set of actions that can be used to mitigate issues in a sandbox or production environment. This feature queries SQL Server directly, so you get query store metrics in near real-time. We have also added an audit trail on the action performed so that you can easily determine who performed the action and when it was performed.

Details

All SQL performance tools in LCS are available under the SQL Insights tab on the Environment Monitoring page for a specific environment. The following tabs are available:

- **Live View** – Shows executing statements and blocking statements. The current SQL Now page that shows performance issues will be replaced with Live View.

- **Queries** – Shows a list of predefined queries that can be used to retrieve metrics on demand. Examples of queries include a current blocking tree, a list of active plan guides, and a list of most expensive queries.

  **IMPORTANT**

  To help guarantee that the query results are returned instantaneously, most of the queries are run synchronously. However, if there is an ongoing performance issue, synchronous query execution might cause a time-out error. To address this issue, a new Use Fast Query option has been added. By default, this option is turned on for most queries. If you receive a time-out error after you run a query, turn the Use Fast Query option off, and then try to run the query again. The query will now run asynchronously.

- **Actions** – Shows a list of predefined actions that should be taken to mitigate issues in the sandbox and production environments. Examples of actions include terminating a blocking statement. Any time that an action is performed, the environment history for an environment will show a record for the action performed. A history record is created only for actions and not when queries are executed.

- **Performance Metrics** – Shows the most expensive queries that were run in the system during the selected period, based on logical I/O, execution count, duration, CPU time, and wait count. This data is queried from the SQL query store. The data is retained for 30 days, and the tool runs its data collection every day at a random time between midnight and 4 AM in the time zone in which your environment is hosted. The last run date and time is visible from your environment details page in Lifecycle Services, under the Monitoring tab in the Last run field. To use the tool, select a period during the last 30 days. When the query results appear, select the bar in the duration chart to highlight where the query falls based on other metrics. On the Statement tab, you can either view the query or download the query execution plan. This feature is not available in self-service environments.
How do I use this feature?

1. Go to your project in LCS and open the environment details page. Select the Environment Monitoring link in the Monitoring section. Select the SQL Insights tab to access this feature.

2. You can navigate to each of the tabs (Live View, Queries, Actions, Performance Metrics, Index, Analysis) to view or query for more information.

3. You have the option to search or export to Excel any of results from the query execution.

4. After you have narrowed down the reason for the performance issue, you can use a predefined action to mitigate the issue.

5. After an action is performed, an entry is made on the Environment History page, which shows the details of the action, the parameters that were passed in, a timestamp, and who triggered the action.

Sample flow

**Scenario:** Users report slow performance when using the system. One issue could be a blocking statement. Blocking by itself is typical in a healthy system and is only a problem when it becomes excessive or starts degrading business activities.

1. Go to the Live View tab and check if there are any blocking statements. If there is a blocking statement, copy the blocking query ID.

2. Open the Queries tab and select the Current Blocking Tree query. This will return the root blocker that is blocking the SQL operation.

3. To resolve the issue, you can either let it run and clear naturally, or end the process for the lead blocker, which will roll work back. Typically, you should only end the lead blocker process if you think that it will not clear naturally (such as a bad query plan), or in situations where a critical process is unable to run and needs to complete immediately.

4. Confirm that it's okay to terminate the statements that are currently being executed.

5. Open the Actions tab and select the End SQL Process action and pass in the root blocker query ID. This will execute a query against the SQL database to terminate the blocking statement.

6. Go to the Queries tab and run Current blocking query to verify if the blocking statement was terminated.

7. You can also check the Environment History page to see details on what process was terminated.

8. To avoid this issue in the future, you should use indexes or plan guides, or turn off lock escalation, or use page locks if processes are blocking each other while operating on different records. If processes are operating on the same records, the only way to avoid blocking is by refactoring or rescheduling the processes to not operate on the same records at the same time.
This topic provides details on each query under the SQL Insights tab on the Environment Monitoring page in Lifecycle Services (LCS) and how they should be used when troubleshooting performance issues. For details about this feature, see Performance troubleshooting using tools in Lifecycle Services (LCS).

Current blocking

Description
Lists any currently blocked queries, and also the SPID that is blocking them, how long they have been blocked, and what resource they are waiting on. This can be used in conjunction with the query to see the blocking tree, which provides a graphical overview of some of the same information. Blocking by itself is normal in a healthy system and is only a problem when it becomes excessive or starts degrading business activities.

Next steps
- Determine which process is blocked, and which process is blocking it and why.
- To resolve blocking, the only two options are to let it run and clear naturally, or to end the lead blocker process, which will roll back work. Generally, the lead blocker should only end in situations where it is not believed that it will clear naturally (such as a bad query plan), or in situations where a critical process is unable to run and needs to be completed immediately.
- To avoid the same blocking in the future, you can use indexes or plan guides, or disable lock escalation and page locks if processes are blocking each other while operating on different records. If processes are operating on the same records, the only way to avoid blocking is by refactoring or rescheduling the processes so that they do not operate on the same records at the same time.

Current blocking tree

Description
Provides a graphical view of the SPIDs and statements that are currently causing blocking or being blocked. This can be used in conjunction with the current blocking query to see more detailed information. Blocking by itself is normal in a healthy system and is only a problem when it becomes excessive or starts degrading business activities.

Next steps
- Determine which process is blocked, and which process is blocking it and why.
- To resolve blocking, the only two options are to let it run and clear naturally, or to end the lead blocker process, which will roll back work. Generally, the lead blocker should only end in situations where it is believed that it will not clear naturally (such as a bad query plan), or in situations where a critical process is unable to run and needs to be completed immediately.
- To avoid the same blocking in the future, you can use indexes or plan guides, or disable lock escalation and page locks, if processes are blocking each other while operating on different records. If processes are operating on the same records, the only way to avoid blocking is by refactoring or rescheduling the processes so that they do not operate on the same records at the same time.

Currently running queries

Description
Provides a list of all queries that are currently in a state of being executed or blocked on this database, and also the total execution and wait times of each query. Queries that have high execution time and low wait time are often indicative of bad query plans. Queries with high wait time and low execution time are indicative of blocking. If relatively fast operations are being run many times, sometimes they can be found by running this query multiple times in a row and looking for commonly occurring queries with fast execution time.

**Next steps**
- If high CPU time is seen, get the query plan for the query, and also see whether other query plans that have been used for this query are more efficient. Consider addressing the issues with a new index, with a change to the query, or, as a last resort, by adding a plan guide.
- If high wait time is seen, view the current blocking and current blocking tree to determine why the query is blocked. This is occasionally addressed by disabling lock escalation or page locks if that is the cause of the blocking. More often, it is addressed by segmenting the work that is being performed to ensure that the same record is not processed by two queries at the same time.

**End SQL process**

**Background**
If a SPID is consuming too many resources and degrading the operation of other processes, it might be beneficial to end the SPID process. This will cause the open transaction to roll back, meaning that data should not be lost, but the process might need to be manually restarted. Note that rollback can also take a long time and consume a lot of resources if the transaction has already performed a lot of work. Therefore, this action should be used with caution.

**Next steps**
- From the blocking tree and other queries, determine which SPID should end.
- Verify that the processing that is being performed by the SPID can end without causing harm to ongoing business operations.
- Provide the SPID number to end, and roll back that operation.

**Removed features**
As stated in Removed or deprecated platform features, some Azure SQL reports and Azure SQL actions have been removed from Lifecycle Services (LCS).

**Removed queries**
The following items have been removed from the Queries tab of SQL Insights in LCS.

<table>
<thead>
<tr>
<th>NAME</th>
<th>REMOVED</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current blocking tree</td>
<td>No</td>
<td>Currently available.</td>
</tr>
<tr>
<td>Current running queries</td>
<td>No</td>
<td>Currently available.</td>
</tr>
<tr>
<td>Current blocking statements</td>
<td>No</td>
<td>Currently available.</td>
</tr>
<tr>
<td>NAME</td>
<td>REMOVED</td>
<td>NOTES</td>
</tr>
<tr>
<td>---------------</td>
<td>---------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>Get indexes</td>
<td>Yes</td>
<td>No longer applicable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Reason</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Manual index management is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>no longer needed as this is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>handled by background platform processes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Details</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The system automatically tunes and manages indexes.</td>
</tr>
<tr>
<td>Get lock details</td>
<td>Yes</td>
<td>No longer applicable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Reason</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The platform is responsible for:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Optimizing the database's workload and handling any blocking that may occur.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Managing intermittent connectivity issues and provides retries to avoid any concerns with such actions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Details</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The platform optimizes workloads and environment to reduce the number of scenarios leading to unresolved process blocking.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Internal monitoring and detection will drive deeper root cause analysis into possible additional scenarios.</td>
</tr>
<tr>
<td>NAME</td>
<td>REMOVED</td>
<td>NOTES</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>-------</td>
</tr>
</tbody>
</table>
| Get list of query ID's | Yes | No longer applicable. **Reason**  
  - With the platform being responsible for query tuning and optimization, insights into individual queries, their plans, and their execution statistics are no longer needed.  
  - Query store information is immensely complex and false interpretations can lead to delays in mitigations and root cause identification. **Details**  
  - The platform will automatically tune and optimize individual queries, removing the need for manual intervention.  
  - Notify Support about performance issues and include high-level details about the areas and timeframes in which slow performance was observed. |
| Get the SQL query plan for a given Plan ID | Yes | Same as above. |
| Get query plans and execution status | Yes | Same as above. |
| Get throttle config | Yes | No longer applicable. **Reason**  
  - Throttling at resource governor-level is no longer applicable for elastic pool. **Details**  
  - This report is no longer applicable. |
<table>
<thead>
<tr>
<th>NAME</th>
<th>REMOVED</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get wait stats</td>
<td>Yes</td>
<td>No longer applicable.</td>
</tr>
</tbody>
</table>

**Reason**

- Database performance is automatically managed by the platform and monitoring of the wait statistics is no longer necessary.

- While invaluable, wait statistics do not provide all information required for root cause analysis and the added complexity can lead to incorrect interpretations and delays in performance mitigations.

**Details**

- The platform will monitor and automatically employ self-healing mechanisms to reduce session wait times.

- Notify Support about performance issues and include high-level details about the areas and timeframes in which slow performance was observed.
The platform targets the top resources consuming queries in several of its automatic tuning operations, meaning the retrieval of these queries are no longer necessary for maintaining system health.

- This query may not show the most concerning queries, just the most expensive query at the period of time. Having this list will not point to concerns in the environment. This is a very expensive query, not targeted for troubleshooting custom queries. It provides more of a general check. It currently fails 10 to 30% of the time.

The platform will monitor and automatically employ self-healing mechanisms to reduce the resource consumption of the most expensive queries.

Notify Support about performance issues and include high-level details about the areas and timeframes in which slow performance was observed.
<table>
<thead>
<tr>
<th>NAME</th>
<th>REMOVED</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current DTU (Database Transaction Unit)</td>
<td>Yes</td>
<td>No longer applicable. Reason</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DTU reports are no longer necessary as the platform monitors all databases and provides adequate resources for all workloads.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Current DTU reports no longer provide an accurate picture of database health. Details</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The platform will monitor databases and automatically optimize the available resources for each workload.</td>
</tr>
<tr>
<td>Current DTU details</td>
<td>Yes</td>
<td>No longer applicable. Reason</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DTU reports are no longer necessary as the platform monitors all databases and provides adequate resources for all customers' workloads.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Current DTU reports no longer provide an accurate picture of database health. Details</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The platform will monitor databases and automatically optimize the available resources for customers' workloads.</td>
</tr>
</tbody>
</table>

**Removed actions**
The following action have been removed from the Actions tab of SQL Insights in LCS.

<table>
<thead>
<tr>
<th>NAME</th>
<th>REMOVED</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>REMOVED</td>
<td>NOTES</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>Create index</td>
<td>Yes</td>
<td>No longer applicable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Reason</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Manual index creation is no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>longer needed as this is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>handled by a background</td>
</tr>
<tr>
<td></td>
<td></td>
<td>platform processes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Details</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A system background process</td>
</tr>
<tr>
<td></td>
<td></td>
<td>will handle this as required.</td>
</tr>
<tr>
<td>Drop index</td>
<td>Yes</td>
<td>No longer applicable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Reason</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Not included in Data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Administration and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Management Service (DAMS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>because of the periodic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nature of Finance and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Operations workloads.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Details</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The system will automatically</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tune as required.</td>
</tr>
<tr>
<td>Rebuilt index</td>
<td>Yes</td>
<td>No longer applicable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Reason</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Manual index creation is no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>longer needed as this is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>handled by background</td>
</tr>
<tr>
<td></td>
<td></td>
<td>platform processes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Details</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A system background process</td>
</tr>
<tr>
<td></td>
<td></td>
<td>will handle this as required.</td>
</tr>
</tbody>
</table>

Manual index creation is no longer needed as this is handled by a background platform processes.

A system background process will handle this as required.

Not included in Data Administration and Management Service (DAMS) because of the periodic nature of Finance and Operations workloads.

The system will automatically tune as required.

Manual index creation is no longer needed as this is handled by background platform processes.

A system background process will handle this as required.
<table>
<thead>
<tr>
<th>NAME</th>
<th>REMOVED</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update statistics</td>
<td>Yes</td>
<td>No longer applicable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Reason</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A platform background process handles index and statistics maintenance.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Details</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The platform is responsible for index and statistics maintenance.</td>
</tr>
<tr>
<td>Query hint optimization</td>
<td>Yes</td>
<td>No longer applicable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Reason</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The platform handles query hint optimization so customers don't have to do manual tuning.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Details</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The platform automatically detects the correct hint and applies it to the queries that need optimization.</td>
</tr>
<tr>
<td>NAME</td>
<td>REMOVED</td>
<td>NOTES</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>---------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>Create a plan guide to add table hints</td>
<td>Yes</td>
<td>No longer applicable.</td>
</tr>
<tr>
<td>Query hint optimization is combined with “Create a plan guide to add table hints”.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reason</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• The platform handles query optimization instead of manual, time-consuming tuning by customers.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• DAMS reduces manual efforts in favor of platform automation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• The platform removes usage of plan guides as they are inefficient and difficult to manage.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Plan guides are being deprecated in favor of forcing hints through query store.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• The platform automatically detects the correct hint and applies it to the queries that need optimization.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create a plan guide to force plan</td>
<td>Yes</td>
<td>Same as above.</td>
</tr>
<tr>
<td>Remove plan guide</td>
<td>Yes</td>
<td>Same as above.</td>
</tr>
<tr>
<td>NAME</td>
<td>REMOVED</td>
<td>NOTES</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------</td>
<td>------------------------</td>
</tr>
<tr>
<td>List of current plan guide</td>
<td>Yes</td>
<td>No longer applicable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Reason</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The platform handles optimization instead of manual, time-consuming tuning by customers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DAMS reduces manual efforts in favor of platform automation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The platform removes usage of plan guides as they are inefficient and difficult to manage.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Details</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Plan guides are being deprecated in favor of forcing hints through query store.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The platform automatically detects the correct hint and applies it to the queries that need optimization.</td>
</tr>
<tr>
<td>End SQL process</td>
<td>No</td>
<td>Continues to be available.</td>
</tr>
</tbody>
</table>
After your subscription has been expired for longer than the 90-day retention period, Microsoft disables the accounts and deletes customer data. Your implementation project in Microsoft Dynamics Lifecycle Services (LCS) is deleted, and any Microsoft-managed environments are deprovisioned and deleted.

Clean up your Azure subscription

After your LCS implementation project is deleted, if you deployed cloud-hosted environments in it via a customer-owned Azure subscription, LCS will no longer have access to the Azure subscription or any of the cloud-hosted environments. However, some resources might remain in your Azure subscription.

Follow these steps to free up the resources, and to remove application permissions in your Azure Active Directory (Azure AD) tenant and each Azure subscription that you previously added to LCS so that you could deploy the cloud-hosted environments.

1. Delete the Azure resources:
   a. In the Azure portal, go to your Azure subscription.
   b. When a cloud-hosted environment is created, it creates a resource group in your Azure subscription. This resource group represents the resources that are created as part of your deployment of the cloud-hosted environment. It has the same name as the deployment. Therefore, it should be easy to find in the list of resource groups. Delete any resource groups that have the prefix DynamicsDeployments-.

   **IMPORTANT**
   If you deployed cloud-hosted environments to more than one Azure region, multiple resource groups might have been created. Be sure to delete all related resource groups.

2. Remove the deployment service application from the subscription:
   a. Sign in to Azure AD via the PowerShell cmdlet.

   ```powershell
   Connect-AzureAD (Using Tenant Administrator account)
   ```

   b. Determine whether the app is still enabled on the Azure AD tenant.

   ```powershell
   Get-AzureADServicePrincipal -Filter "AppId eq 'b96b7e94-b82e-4e71-99a0-cf7fb188acea'"
   ```

   c. If the preceding command returns an object, the app is currently enabled on the tenant, and it might still have access to the subscription. Remove the app from the tenant.

   ```powershell
   $DDSOBJECTId=$(Get-AzureADServicePrincipal -Filter "AppId eq 'b96b7e94-b82e-4e71-99a0-cf7fb188acea'".).ObjectId
   Remove-AzureADServicePrincipal -ObjectId $DDSOBJECTId
   ```
d. Verify that the app has been removed.

```
Get-AzureADServicePrincipal -Filter "AppId eq 'b96b7e94-b82e-4e71-99a8-cf7fb188acea'"
```

Related topics

- Data retention, deletion, and destruction in Microsoft 365
- Subscriptions, LCS projects, and Azure Active Directory tenants FAQ
The Microsoft Dynamics 365 Translation Service (DTS) is hosted in Microsoft Dynamics Lifecycle Services (LCS). It's designed to enhance the experience for partners and independent software vendors (ISVs) when they translate their solutions or add a new language for supported Dynamics products.

If you're interested in learning the basics and best practices of DTS, consider completing the Translate Dynamics 365 apps and documentation with Dynamics 365 Translation Service module on Microsoft Learn.

DTS uses product-specific machine translation (MT) models that are custom-trained for Microsoft General Availability (GA) languages to maximize the quality of the translation output. DTS also supports translation recycling from the linguistic assets of Microsoft Dynamics and partners/ISVs. Therefore, identical strings are translated one time and then consistently reused.

The following illustration shows, at a high level, how the service works.

![Service Workflow Diagram](image)

**Recycling existing translations**

Existing linguistic assets can be recycled only when the assets are uploaded in a zip file that contains translation memory (TM) files that use Localization Interchange File Format (XLIFF). For more information, see Translation memory files.

**Custom-trained MT model**

DTS uses a Microsoft Translator service and a custom translator to customize Microsoft Translator's advanced neural machine translation for Microsoft Dynamics products. The custom-trained MT model can be used for requests, provided that the source or target language is English and partners upload XLIFF TM files that contain more than 10,000 translation units (TUs). (A TU typically contains a source string, translation, state, state qualifier, and note.) In those cases, DTS creates a custom-trained MT model that is specific to the translation request that the XLIFF TM files are submitted for.

**Supported products**

DTS currently supports the following product versions.
<table>
<thead>
<tr>
<th>PRODUCT NAME</th>
<th>VERSIONS</th>
<th>SUPPORTED FORMAT FOR USER INTERFACE FILES</th>
<th>SUPPORTED FORMAT FOR DOCUMENTATION FILES</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Dynamics AX 2012</td>
<td>All versions</td>
<td>.ktd, .ald</td>
<td>.docx</td>
<td>.txt is the specific label format and .html is the custom help solution format.</td>
</tr>
<tr>
<td>Dynamics 365 Finance and Operations apps</td>
<td>All versions</td>
<td>.label.txt</td>
<td>.docx, .html</td>
<td>.txt is the specific label format and .html is the custom help solution format.</td>
</tr>
<tr>
<td>Microsoft Dynamics 365 Commerce</td>
<td>All versions</td>
<td>.label.txt</td>
<td>.docx</td>
<td>.txt and .xml are the NAV specific format, and .xlf is the Business Central extension resource format.</td>
</tr>
<tr>
<td>Microsoft Dynamics CRM</td>
<td>All versions</td>
<td>.resx</td>
<td>.docx</td>
<td>.txt and .xml are the NAV specific format, and .xlf is the Business Central extension resource format.</td>
</tr>
<tr>
<td>Microsoft Dynamics NAV</td>
<td>All versions</td>
<td>.etx, .stx, .resx, .txt, .xml, .xlf</td>
<td>.docx</td>
<td>.txt and .xml are the NAV specific format, and .xlf is the Business Central extension resource format.</td>
</tr>
</tbody>
</table>

### Accessing DTS

You can access DTS in two places in LCS:

- From the LCS home page
- From within an LCS project

**Accessing DTS from the LCS home page**

Sign in to LCS, and scroll to the right side of the page. Expand the tiles waffle, and then select the Translation service tile to open the dashboard view for DTS.

**Accessing DTS from within an LCS project**

Create a new project, or open an existing project. On the project dashboard, in the More tools section, select the Translation service tile. Alternatively, on the project dashboard, select the Menu button, and then select Translation service.

**Accessing DTS from the LCS home page vs. accessing it from within an LCS project**

When you access DTS from the LCS home page and create a translation request, you can select the product that is used for the request. To add more requests that use different products, you can just change the product selection. You don't have to close the service and open a different translation project. This option is convenient when you work on multiple product translation projects.

The following illustration shows an example of the DTS dashboard that you open from the LCS home page. This dashboard shows all requests that users from your organization have made from the home page (outside an LCS project).
Because an LCS project is always tied to a product, any translation request that you submit from a project automatically carries the product type and version information from the project. You can’t select a different product for the request.

The following illustration shows an example of the DTS dashboard that you open from within an LCS project. Only the project owner and users who have access to the project will be able to see these requests. Therefore, this option is useful when you work with a group of people on one product translation project in LCS.

In both situations, users who can view the request will have read access. However, to regenerate the request, they must take ownership of it in the dashboard.

Glossary

<table>
<thead>
<tr>
<th>TERM</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>


<table>
<thead>
<tr>
<th>TERM</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>XLIFF</td>
<td>XML Localization Interchange File Format. XLIFF is an XML-based format. It was created to standardize the way that localizable data is passed between tools during a localization process, and to serve as a common format for files that are used by computer-aided translation (CAT) tools.</td>
</tr>
<tr>
<td>Microsoft GA languages</td>
<td>General availability of the Microsoft-produced languages. The list varies, depending on the product.</td>
</tr>
<tr>
<td>TU</td>
<td>Translation unit. A TU typically contains a source string, translation, state, state qualifier, and note.</td>
</tr>
</tbody>
</table>

For more information about how to use DTS, see [Translate user interface files](#) and [Translate documentation files](#).
This topic provides information about how to translate a user interface (UI) file for Microsoft Dynamics products or solutions.

For more information about the Microsoft Dynamics 365 Translation Service, see Dynamics 365 Translation Service overview. For information about how to translate a documentation file, see Translate documentation files.

Create a translation request

1. In Microsoft Dynamics Lifecycle Services (LCS), on the DTS dashboard, select Add to create a new translation request.

2. Enter the required information for the request.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request name</td>
<td>Enter a name for the request.</td>
</tr>
<tr>
<td>File type</td>
<td>Select User Interface.</td>
</tr>
<tr>
<td>Product name</td>
<td>Select a product name. If you accessed DTS from within an LCS project, this field is automatically filled in and is read-only.</td>
</tr>
<tr>
<td>Product version</td>
<td>Select a product version. If you accessed DTS from within a LCS project, this field shows the default product version information from the project. However, you can select a different version.</td>
</tr>
</tbody>
</table>
**Translation source language, Translation target language**

Select the set of source and target languages to translate from and to. If your business requires that multiple target languages be translated for the same source language, you can select all the target languages in one request. Select each target language by using the checkbox next to the language's name. This approach helps you save time and also lets you track the status of all the target language translations in one request. The fields list all the languages that are supported for the selected product name and version. Language names that are shown in **bold** are General Availability (GA) languages for Microsoft Dynamics products. Therefore, product-specific machine translation (MT) models are available in those languages, and the MT model is trained on the terminology for Microsoft Dynamics. For non-GA languages, the MT model uses the general domain training.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation target language</td>
<td>Select the set of source and target languages to translate from and to. If your business requires that multiple target languages be translated for the same source language, you can select all the target languages in one request. Select each target language by using the checkbox next to the language's name. This approach helps you save time and also lets you track the status of all the target language translations in one request. The fields list all the languages that are supported for the selected product name and version. Language names that are shown in <strong>bold</strong> are General Availability (GA) languages for Microsoft Dynamics products. Therefore, product-specific machine translation (MT) models are available in those languages, and the MT model is trained on the terminology for Microsoft Dynamics. For non-GA languages, the MT model uses the general domain training.</td>
</tr>
</tbody>
</table>

3. Select **Create**. Verify the request details were selected correctly and then click **Yes** to continue.
NOTE
To take advantage of the product-specific model that is trained on Microsoft Dynamics linguistic assets, you must select **English – United States** as either the source language or the target language. Here is an example.

<table>
<thead>
<tr>
<th>TRANSLATION SOURCE LANGUAGE</th>
<th>TRANSLATION TARGET LANGUAGE</th>
<th>MT MODEL THAT IS USED</th>
</tr>
</thead>
<tbody>
<tr>
<td>English – United States</td>
<td>Japanese</td>
<td>Product-specific trained MT model</td>
</tr>
<tr>
<td>Japanese</td>
<td>English – United States</td>
<td>Product-specific trained MT model</td>
</tr>
<tr>
<td>German</td>
<td>Japanese</td>
<td>Generic MT model</td>
</tr>
</tbody>
</table>

Upload files

Select the plus sign (+) in each section to open the **File upload** page.

**Upload the files to translate (Required)**

Create one zip file that contains all the UI files in the source language that you want to translate from. The zip file can include different file types, provided that the file types are supported for the product. For more information about supported file types, see [Supported products](#). Note that DTS doesn't change the source files that you upload. The source files are only used to create files in the corresponding target languages you requested.

**Upload XLIFF translation memory files (Optional)**

If you have XLIFF TM files from a previous UI translation request, or if you used the Align tool to create an XLIFF TM, create a zip file that contains all TM files before you upload them. Strings that match are then recycled to help guarantee consistency between product versions. For more information about XLIFF TMs, see [Translation memory files](#).

If you created the translation request for multiple target languages, you must select which target language the TM file is for.

You have an option to create a custom **MT model** that is trained with the translation memory file that you're
After translation is completed, if you use this option, the request might take longer to be completed. You must select either Yes or No before you can continue with the TM file upload.

After you've finished uploading files, select Submit to start the translation process.

After you submit the request, a new request ID is created on the DTS dashboard. If you submitted the request for multiple target languages, the status of each target language is shown on a separate line that has the same request ID. If you select a line on the dashboard, the dashboard page will be extended to the right to show a summary of the request information.

To see the request status, click a request ID link on the dashboard. The Request status tab shows the source files list you uploaded with the summary of the request information.

Note that the processing time depends on the number of requests that are in the DTS queue and the word count in the source files that you submit.

- UI translation requests that don't have an XLIFF TM can be completed in a few minutes, depending on the file size.
- If a UI translation request does have an XLIFF TM, the time that is required depends on the type of MT model:
  - Creation of a custom MT model requires two to three business days.
  - If you're using a generic MT model, requests can be completed in a few minutes, depending on the file size.

After translation is completed

When your translation request has been processed, you will receive an email notification from DTS. You can then view the result on the Request output tab of the Request details page.
For UI translation requests, two types of output file are available after the translation process is completed.

- **For translation review** – Download the XLIFF file to review and, as required, edit the translations. The file shows the source and target languages side by side.
- **Translated native format** – Download this file if you don't intend to review or edit the translations. *Native format* means that the file is in the same format as the source file that you submitted.

Click an individual file link or the download links to download a single file, all files for one target language, or all files for all target languages in one zip for convenience.

**Review and edit the translations in the XLIFF file**

We recommend that you review and edit the translations in the XLIFF file that DTS provides, to verify that the translation output meets your product's quality standards. For more information about how to edit the XLIFF file, see Translation memory files.

**Regenerate output files**

When you've finished reviewing and editing the translation files in XLIFF, you must regenerate the translated native format files next. You can then apply the latest translations (that is, your edited versions of the translations) to the UI files in the target language. You can regenerate any number of files from the output files set per target language.

1. Click the **Regenerate** icon next to the target language section. The **File upload** slider will open.
2. Zip the edited XLIFF files, and then click **Upload**. Don't change the XLIFF file name that DTS originally provided.
3. In the prompt, confirm the upload.
4. The **Request output** tab promptly refreshes the content. Expand the target language node you just regenerated to verify the **Modified** timestamp and then download the updated output files.
You can repeat the regeneration process as many times as you require.
This topic explains how to translate a documentation file for Microsoft Dynamics products and solutions.

Create a translation request

1. In Microsoft Dynamics Lifecycle Services (LCS), on the DTS dashboard, select Add to create a new translation request.

You can open the DTS dashboard either from the LCS home page or from within a project. For more information, see Accessing DTS.

2. Enter the required information for the request.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request name</td>
<td>Enter a name for the request.</td>
</tr>
<tr>
<td>File type</td>
<td>Select Documentation.</td>
</tr>
<tr>
<td>Product name</td>
<td>Select a product name. If you accessed DTS from within an LCS project, this field is automatically filled in and is read-only.</td>
</tr>
<tr>
<td>Product version</td>
<td>Select a product version. If you accessed DTS from within an LCS project, this field shows the default product version information from the project. However, you can select a different version.</td>
</tr>
</tbody>
</table>
3. Select **Create**.
NOTE
To take advantage of the product-specific model that is trained on Microsoft Dynamics linguistic assets, you must select English – United States as either the source language or the target language. Here is an example.

<table>
<thead>
<tr>
<th>Translation Source Language</th>
<th>Translation Target Language</th>
<th>MT Model That Is Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>English – United States</td>
<td>Japanese</td>
<td>Product-specific trained MT model</td>
</tr>
<tr>
<td>Japanese</td>
<td>English – United States</td>
<td>Product-specific trained MT model</td>
</tr>
<tr>
<td>German</td>
<td>Japanese</td>
<td>Generic MT model</td>
</tr>
</tbody>
</table>

Upload files

Select the plus sign (+) in each section to open the File upload page.

Upload files to translate (Required)
Currently, only files in Microsoft Word (.docx) format are accepted for translation. Create a zip file that includes all the .docx files in the source language that you want to translate from. You can upload only one zip file. Note that DTS doesn’t change the source files that you upload. The source files are only used to create files in the corresponding target languages you requested.

Upload XLIFF or TMX translation memory files (Optional)
If you have a TM in Translation Memory eXchange (TMX) format from a previous DTS request, and/or if you have a XLIFF TM from UI file translation, you can attach those TMs so that they can be recycled in the new document that you’re submitting. Create a zip file that includes all the TM files. You can upload only one zip file. If you created the translation request for multiple target languages, you must select which target language the TM file is for.

With the translation memory file you are providing, you have an option to decide whether you want to create a custom MT system trained with it. This option may take longer time to complete the request. You must choose
Yes or No to be able to continue with the TM file upload.

After you've finished uploading file, select **Submit** to start the translation process.

After you submit the request, a new request ID is created on the DTS dashboard. If you submitted the request with multiple target languages, you will see each target language status is displayed in a separate line with the same request ID. Selecting a line on the dashboard will extend the dashboard window to the right to show the request summary information.

To view the request status, select a request ID link on the dashboard. The **Request status** tab shows the list of source files that you uploaded, together with a summary of the request information.

Note that the processing time depends on the number of requests that are in the DTS queue and the word count in the source files that you submit.

**After translation is completed**

When processing of your translation request is completed, you receive an email notification from DTS. You can then view the result on the **Request output** tab of the request details page. If your request was submitted for multiple target languages, there may be a difference between languages when each language process is done. Expand each language name to see the status.
For documentation translation requests, three types of output file are available after the translation process is completed:

- **For translation review** – Download this file to review and edit the translated document strings in a table view. The file shows the source and target languages segments side by side.

- **Translated native format** – Download this file if you don't intend to review or edit the translations, but intend to use the translated file as it is. This file has the same formatting style (title, headings, tables, and so on) as the source .docx file that you submitted, and it's ready to be used.

- **Translation memory** – Download this file to recycle these translations the next time that you submit a translation request that uses a newer version of the source document.

**Review and edit the translations**

DTS provides the translation review file in .docx format. You can download the file from the Request output tab of the request details page and open it in Word. The file provides a convenient table view, as shown in the following illustration. Therefore, you can easily compare the text in the source and target languages side by side. After you've finished reviewing the file, you must save it and upload it back to DTS to generate the updated .docx file output in the original formatting style that you submitted.

<table>
<thead>
<tr>
<th>Segment ID</th>
<th>Segment status</th>
<th>Source segment</th>
<th>Target segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Draft (0%)</td>
<td>&lt;109&gt;&lt;117&gt; &lt;110&gt; &lt;103&gt; Pre Go-live checklist &lt;/109&gt; &lt;/110&gt; &lt;/117&gt;</td>
<td>&lt;109&gt;&lt;117&gt;&lt;110&gt; &lt;103&gt; Avant mise en service liste de vérification &lt;/109&gt; &lt;/110&gt; &lt;/117&gt;</td>
</tr>
<tr>
<td>2</td>
<td>Draft (0%)</td>
<td>Customer:</td>
<td>Client :</td>
</tr>
<tr>
<td>3</td>
<td>Draft (0%)</td>
<td>Partner:</td>
<td>Partenaire :</td>
</tr>
<tr>
<td>4</td>
<td>Draft (0%)</td>
<td>LCS project name:</td>
<td>Nom du projet LCS :</td>
</tr>
<tr>
<td>5</td>
<td>Draft (0%)</td>
<td>LCS ID:</td>
<td>ID DE LETTRES DE CRÉDIT :</td>
</tr>
<tr>
<td>6</td>
<td>Draft (0%)</td>
<td>Go-live date:</td>
<td>Date de mise en service :</td>
</tr>
<tr>
<td>7</td>
<td>Draft (0%)</td>
<td>DD-MM-YYYY</td>
<td>AAAA-MM-JJ</td>
</tr>
<tr>
<td>8</td>
<td>Draft (0%)</td>
<td>Dynamics 365 for Finance and Operations, Enterprise edition customers should complete a Pre-Go-live review with the Microsoft FastTrack team before requesting their production environment.</td>
<td>Dynamics 365 pour Finance and Operations, clients Édition entreprise doit remplir une révision live donné avant avec l'équipe Microsoft FastTrack avant de demander son environnement de production.</td>
</tr>
</tbody>
</table>

When you edit the .docx review file, note of the following guidelines:

- Edit only the text in the Target segment column.
- Don't add or remove rows.
- Don't change the order of the rows or columns.
- Don't add or remove the red tags. Most red tags represent formatting and styles.
- If you must move the red tags, be careful that you don't switch a start tag (for example, &lt;116&gt;) and its end tag (&lt;/116&gt;).

**Regenerate output files**

When you've finished reviewing and editing a .docx review file, you must regenerate the output file in the source document style. You can then apply the latest translations (that is, your edited versions of the translations) to the documentation files in the target language.

1. Click the Regenerate icon next to the target language section. It brings in the File upload slider.
2. Zip the edited .docx files, and then select Upload. Don't change the file names that DTS originally provided for the .docx review file.
3. You're prompted to confirm the upload action.
4. Once the regenerate is processed, the Request output tab refreshes the content. This process may take some time to complete.
You can repeat the regeneration process as many times as you require.

For more information about the Microsoft Dynamics 365 Translation Service (DTS), see Dynamics 365 Translation Service overview. For information about how to translate a user interface (UI) file, see Translate user interface files.
Microsoft Dynamics 365 Translation Service (DTS) uses a bilingual XML Localization Interchange File Format (XLIFF) file to store pairs of source languages and target languages. Because XLIFF is based on XML, you can open XLIFF files in any text editor. However, we recommend that you use XLIFF editors that are specifically designed to work with this format. For example, you can use the free Microsoft Multilingual Editor that is available in the Multilingual App Toolkit (MAT).

In DTS, you can obtain an XLIFF translation memory (TM) in two ways:

- **Run the Align tool** – When you have files that were previously translated, and you also have corresponding source files, you can use the Align tool to create an XLIFF TM. For more details, see the Creating a translation memory section later in this topic.
- **Complete a translation request** – When a DTS translation request is completed, it provides the XLIFF TMs as part of the request output. You can then use the files the next time that you submit a new translation request that includes the updated source files.

XLIFF files contain a series of translation units (TUs) that are extracted from the source files. The following illustration shows an example of a TU.

```
<trans-unit id="PRO39" translate="yes" xml:space="preserve">
  <source>Rebate</source>
  <target state="translated" state-qualifier="exact-match">リベート</target>
</trans-unit>
```

The following illustration shows the same TU (highlighted in blue) in the Multilingual Editor.

**State**

Each translation in the XLIFF file is associated with a state value. The state value that DTS assigns to each translation depends on the way that the string is translated. When an XLIFF TM is created by using the Align tool, all translations are marked as **Translated**, because the aligned TUs are produced from known good translations, such as a previous product version.
However, when the XLIFF files are generated through a translation request, two types of state values can be used:

- **Needs Review** – The string has been machine-translated.
- **Translated, Final, or Signed off** – The string has been recycled. The state value is inherited from the XLIFF TM.

During the post-editing process, you can immediately identify the strings that are marked as Needs Review. After you’ve finished reviewing those strings, you should mark them as Translated, Final, or Signed off, so that they can be used for recycling. Translations that are marked as Needs Review aren’t included for recycling. Inherited state values for recycled strings are helpful, because you won’t have to review the same string (that is, a string that has the same ID) again.

**Creating a translation memory**

If you have files that were previously translated, you can recycle the translated files for a newer version of the source files by creating a TM that uses XLIFF.

1. On the DTS dashboard, select the **Align** button to start the Align tool.

   ![Align tool image]

   **NOTE**
   
   - The Align tool currently supports only user interface (UI) files.
   - To start the Align tool, you might have to explicitly allow pop-up windows in your browser.

2. On the **Align** page, select the source language, the target language, and the files to align.

3. Select **Align** to complete the alignment. When the alignment is completed, a message summarizes the results.
To create the best XLIFF TM, make sure that the following conditions are met:

- Both the source file and the target file have the same number of resources.
- The resources are in the same order in both the source file and the target file.
- There are no empty strings. The following illustration shows examples of empty strings in the source and the target.

Empty strings are inherited by the XLIFF TM. If a **Rebate** string in the source has an empty string in the
Editing an XLIFF translation memory

We recommend that you use the free Multilingual Editor, or another XLIFF editor, to review and edit the translations in the XLIFF file that DTS provides. At a minimum, you should review the translations to verify that the translation output meets your product’s quality standards.

When you open an XLIFF file in the Multilingual Editor, it resembles the following illustration. If you encounter an error when you open the file, ignore the message, and select the **Strings** tab in the lower-left corner of the window.

Notice that there is a circle near the beginning of each line. The color of the circle indicates the state of the translation. DTS automatically assigns these states, depending on where the string came from.

- **Red circle** – The string was machine-translated. DTS assigns the **Needs Review** state.

Although the Align tool can resolve some of these issues, it’s easier if you prevent them before you see unexpected results in the output.

Review the aligned XLIFF file before you use it as a TM. TUs that have been reviewed should be marked as **Final** or **Signed off**, so that they aren’t mistaken for unreviewed TUs.
NOTE
The state value that is shown might differ slightly, depending on the XLIFF editor that you’re using.

- Yellow, green/yellow, or green circle – The string was recycled. DTS inherited the state from the XLIFF TM that was used in the request.

To verify the translations, you can apply a filter to show only strings that are in the Needs Review state.

Strings that have been reviewed should be marked as Translated, Final, or Signed off, so that they can be used for recycling. Translations that are marked as Needs Review aren’t included for recycling.

After you’ve finished editing the XLIFF TM, remember to have DTS regenerate the refreshed output file in the source format. For more information about how to regenerate the file, see Translate user interface files.
Microsoft Power Platform provides a suite of capabilities for Dynamics 365 applications via the Power Platform admin center. Today, Finance and Operations apps are not managed by the Power Platform admin center. However, over time more and more management capabilities will be migrated from Microsoft Dynamics Lifecycle Services (LCS) over to the admin center. In the interim, customers will be able to unlock features, such as dual-write functionality, virtual entities, add-ins, and more via Microsoft Power Platform integration functionality in LCS.

Prerequisite reading

To understand the architecture of Microsoft Power Platform, Dataverse, dual-write, and virtual entities for Finance and Operations apps, you must understand how they work. Therefore, the following documentation is a prerequisite:

- Administer Power Platform
- What is Dataverse?
- Tables in Dataverse
- Entity relationships overview
- Create and edit virtual tables that contain data from an external data source
- What is Power Apps portals?
- Overview of creating apps in Power Apps

Tools and services unlocked with Microsoft Power Platform integration

Together, virtual entities, dual-write, business events, and data events make up the shared data layer for the convergence of Finance and Operations apps and the Dataverse platform. They are complementary technologies that are intended to work together.

Virtual entities enable scenarios where access to Finance and Operations data from Microsoft Power Platform or native Dataverse apps is required. You can query that data, bind forms to it, and generally use the full power of Microsoft Power Platform against the full breadth of Finance and Operations apps. Data isn’t copied between systems. Instead, it’s accessed directly through the standard virtual entity infrastructure that Microsoft Power Platform technologies can already bind to. For more information, see Virtual entities overview.

Business events let you use Microsoft Power Platform to respond to events that are occurring in Finance and Operations apps. These events occur when a process is run in the application with business logic. Business events can be raised from any app, including Finance and Operations apps, and can be handled by Microsoft Power Platform business logic. This handling will often include querying or interacting with additional data through either native entities or virtual entities.

Data events, similar to business events, enable external applications to receive notifications from Finance and Operations apps when events occurs. Data events occur when there is a change to a record in the application data. External systems can react to notifications when a create, update, or delete (CUD) operation occurs in the data.
For a subset of scenarios, data must be physically copied between Finance and Operations apps and native Dataverse entities. These scenarios are for overlapping entities that already have a large amount of bound logic in both native Dataverse apps and Finance and Operations apps, so that the data must reside in the local database of each type of app. Although the number of these entities is relatively small, it includes some of the most important entities, such as Account/Customer, Company, Product, and Sales order. For these scenarios, dual-write enables near-real-time synchronous copying of data. This capability enables existing apps to continue to operate against local data, as designed, and also ensures that the corresponding overlapping entity is kept in sync. For more information, see the Dual-write home page.

Together, virtual entities, dual-write, business events, and data events let you build apps and business processes that span the boundaries between Finance and Operations apps and native Dataverse apps. Most apps and business processes will use either a combination of these three parts of the shared data layer or all of them. As always, extension and customization should reduce the amount of data that is copied between databases as much as possible, and should also optimize for the best possible user experience when these tools are used.

**Add-ins functionality**

Add-ins provide a way to extend the functionality of Finance and Operations apps. All add-ins are installed and managed via Lifecycle Services on the environment details page for sandbox and production-type environments. The metadata regarding which add-ins are installed and the configuration options for each add-in are stored in the Microsoft Dataverse database that is provisioned as part of the Microsoft Power Platform integration. Some add-ins also store business data in the Dataverse database. To learn more about available add-ins, see Add-ins overview.

**Typical scenarios and patterns that use dual-write**

Here are some typical scenarios that use dual-write.

**Customer service representatives can facilitate a change of address for Finance and Operations customers**

A customer relocates and wants to change their billing and shipping address information. This customer contacts a customer service representative and requests a change of address. The customer service representative takes the call and changes the customer’s billing and shipping address information.

<table>
<thead>
<tr>
<th>DECISION</th>
<th>INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is real-time data required?</td>
<td>Yes</td>
</tr>
<tr>
<td>Peak data volume</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Frequency</td>
<td>Ad hoc</td>
</tr>
</tbody>
</table>

**Recommended solution**

This scenario that involves near-real-time data synchronization is best implemented by using dual-write.

1. The customer’s information is sourced in a Finance and Operations app.
2. A customer calls customer service and asks to change their billing and shipping address information.
3. A customer service representative retrieves the customer’s record in Dynamics 365 Customer Service.
4. The customer service representative updates the billing and shipping addresses, and saves the data.
5. The new billing and shipping addresses are synced back to the Finance and Operations app in real time.

**Sales representatives can change customer credit limits without signing in to a Finance and Operations app**

A customer has a credit limit of $2,000 and wants to increase it to $5,000. This customer calls and requests the increase. The ticket is assigned to the sales department. The head of sales reviews the request, reviews the customer’s payment history, and determines that the customer is eligible for an increased credit limit. The head of sales approves the request and responds to the ticket. The customer receives an email about the approval of
the $5,000 credit limit.

<table>
<thead>
<tr>
<th>DECISION</th>
<th>INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is real-time data required?</td>
<td>Yes</td>
</tr>
<tr>
<td>Peak data volume</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Frequency</td>
<td>Ad hoc</td>
</tr>
</tbody>
</table>

**Recommended solution**

This scenario is best implemented by using dual-write.

1. A customer calls and wants to increase their credit limit from $2,000 to $5,000.
2. A customer support representative creates a ticket in Dynamics 365 Customer Service.
3. The ticket is assigned to the sales unit.
4. A sales representative from the sales unit reviews and approves the request.
5. The customer's credit limit is increased to $5,000 in Dynamics 365 Sales.
6. The credit limit in the Finance and Operations app is updated to $5,000.
7. The sales representative responds to the ticket and resolves it.
8. The customer receives an email about the increased credit limit.
The integration of Finance and Operations apps with Microsoft Power Platform can be enabled when you create a new Finance and Operations apps environment in Microsoft Dynamics Lifecycle Services (LCS). Alternatively, Microsoft Power Platform can be enabled in an existing Finance and Operations apps environment. For both options, you must complete the setup prerequisites.

Prerequisites for setting up the Microsoft Power Platform integration

The following list describes the prerequisites for setting up the Microsoft Power Platform integration.

- Make sure that at least one gigabyte (GB) of Microsoft Power Platform database storage capacity space is available for your tenant. If this space isn't available, the setup will fail. View your capacity in the Power Platform admin center.

- Identify your Finance and Operations apps environment administrator. You can find that information in the Environment details section.

- Validate the governance policy of your Microsoft Power Platform environment. To do this validation, you must have either the Global administrator role or the Power Platform administrator role.
  
  1. Sign in to the Power Platform admin center.
  2. Select Settings in the upper-right corner of the page to open the Power Platform settings pane.
For organizations that don’t allow everyone to create Microsoft Power Platform production environments, the Finance and Operations apps environment administrator account for your environment must be added to one of the following Microsoft Power Platform admin roles. To make this change, you must have the **Global administrator** role.

- Global admins
- Dynamics 365 admins
- Microsoft Power Platform admins

**NOTE**

The preceding roles might provide more permissions than the Finance and Operations apps environment administrator account requires. Therefore, a more limited role for this integration will eventually be added to Azure Active Directory (Azure AD). The new role won’t require any of the preceding roles. If you want to keep the administrator that has the least privileges, you can temporarily grant one of the preceding roles. Then, after the Microsoft Power Platform integration is set up, remove that role.

All users who create Microsoft Power Platform environments must be licensed. The Microsoft 365 admin center should be used to apply the **Dynamics 365 Unified Operations Plan** license, the **AX Enterprise** license, or an application-specific license such as **Dynamics 365 Finance** to the Finance and Operations apps environment administrator account.
Enable integration during environment deployment

When you set up a new Finance and Operations apps environment in LCS, the deployment wizard includes several sections where you can set values. One of those sections is named **Power Platform Integration**.

Follow these steps to configure the **Power Platform Integration** section.

1. Set the **Configure Power Platform Environment** option to **Yes**. Several additional settings become available.

2. In the **Power Platform template** field, select one of the following values:
   - **Dynamics 365 Standard** – This basic template is applicable to all Finance and Operations apps environments. Select this value if you don’t require a more specific template.
   - **Project Operations** – This template is specific to the Dynamics 365 Project Operations scenario. This value is available only if your tenant has licenses and entitlement for Project Operations.

3. If you’re deploying a DevTest or cloud-hosted environment, the **Environment Type** field is available. There, you can select the type of Dataverse environment that is created and linked. Otherwise, by default, the environment type is set to **Sandbox** for Tier 2 through Tier 5 acceptance test environments and **Production** for production environments.

4. Select **Agree** to agree to the terms and conditions of the integration.

**IMPORTANT**

The **language** and **currency** values of the Dataverse environment that is created and linked to your Finance and Operations apps environment are automatically determined, based on the physical address of your Azure AD tenant. For example, if the address is in Redmond, Washington, USA, the language will be English by default, and the currency will be US dollars (USD).

If you require values that differ from the default values, contact Microsoft support. We can help link an existing Dataverse environment that you manually provision to the Finance and Operations apps environment. Eventually, fields for the language and currency will be added as setup options, so that you can manually set them or accept the default values.

Enable integration after environment deployment

If the Microsoft Power Platform integration isn’t enabled during deployment of the Finance and Operations apps environment, you can enable it in LCS after deployment. To do the setup after the Finance and Operations apps environment has been deployed, follow these steps.
1. After the Finance and Operations apps environment has been deployed through LCS, open the **Environment details** page in LCS.

2. In the **Power Platform integration** section, select **Setup**.

   ![Lifecycle Services](image.png)

   **Manage environment**

   **Power Platform Integration**

   Expand your environment’s capabilities through Power Platform integration. Unlock features such as Dual-write, add-ins, and all the capabilities of Microsoft Dataverse.

   **POWER PLATFORM ENVIRONMENT INFORMATION**

   - **Setup**

   **ENVIRONMENT ADD-INS**

   Power platform environment must be setup to install add-ins.

3. In the **Power Platform environment setup** dialog box, agree to the terms and conditions, and then select **Setup**.

   ![NOTE](image.png)

   **NOTE**

   A Dataverse-based environment will now be provisioned in the Power Platform admin center. The environment typically requires 1 GB of database storage capacity and will have the same name as your Finance and Operations apps environment. Dual-write platform-level components will be installed, but dual-write application components won’t be set up or enabled. Those actions are separate.

4. When you receive a message that states that the Microsoft Power Platform environment is being provisioned, select **OK**.

   The **Power Platform integration** section of the **Environment details** page now shows a message that states that the Microsoft Power Platform environment is being provisioned.

5. After a few minutes, refresh the **Environment details** page.

6. In the **Power Platform integration** section, notice that the value of the **Status** field is **Environment setup is in progress**.

   Typically, the setup takes between 60 and 90 minutes.

   After the Dataverse environment is provisioned, the **Install a new add-in** and **Dual-write application** buttons become available in the **Power Platform integration** section.
Enable integration with an existing, selected Microsoft Power Platform environment

When you enable the Microsoft Power Platform integration for a Finance and Operations apps environment in LCS, either during or after deployment, the process creates a new Dataverse-enabled Microsoft Power Platform environment and links the Finance and Operations apps environment to the new Microsoft Power Platform environment. However, you might want to enable integration by linking your Finance and Operations apps environment to an existing Microsoft Power Platform environment rather than the environment created automatically during deployment. The option to select the Power Platform environment with which to enable the Power Platform integration isn't currently available in LCS.

How the Power Platform integration is enabled for existing environments depends on the number of Power Platform environments that are linked to the Finance and Operations apps environment. Before you enable the Power Platform integration, there are multiple ways in which a Finance and Operations apps environment can...
be considered linked to a Power Platform environment:

- **Deployment link**: During the deployment of a new Finance and Operations environment, even when the option to enable the Power Platform integration is not selected, a new Power Platform environment is created and linked to the Finance and Operations environment. You can see this link in the Finance and Operations URL field of the environment details for the new Power Platform environment in the Power Platform admin center.

- **Dual-write**: The option is available in the dual-write configuration to create a link to a Dataverse environment in any Power Platform environment on the tenant.

- **Virtual entities**: The virtual entity configuration in the Power Platform environment allows you to select the Finance and Operations environment.

These three configurations won't necessarily link the Finance and Operations apps environment to the same Power Platform environment. There are two scenarios for enabling the Power Platform integration with an existing Microsoft Power Platform environment:

- **Single Power Platform environment**: In this scenario, the Finance and Operations apps environment has matched links to a single Power Platform environment. If dual-write and/or virtual entities have been configured for the Finance and Operations apps environment, they are configured to link to the same Power Platform environment created during deployment of the Finance and Operations apps environment.

- **Multiple Power Platform environments**: There is a linking mismatch among existing links between the Finance and Operations apps environment and more than one Power Platform environment.

**Finance and Operations apps connected to a single Microsoft Power Platform environment**

If the Finance and Operations apps environment is configured with links to a single Microsoft Power Platform environment, these environments are identified as a one-to-one linking. For these environments, in Finance and Operations apps version 10.0.22 (Platform update 46) a one-to-one linked environment will automatically be updated to enable the full Microsoft Power Platform integration for the Finance and Operations apps environment.

In version 10.0.22, you can verify that the Microsoft Power Platform integration was automatically enabled by viewing the Power Platform Integration section of the Environment details page for the Finance and Operations apps environment in LCS. If the integration was successfully enabled, the Environment name field will show the name of the integrated Microsoft Power Platform environment, and the Status field is set to Setup completed successfully.

**Finance and Operations apps connected to multiple Microsoft Power Platform environments**

If a Finance and Operations apps environment has been manually linked to multiple Microsoft Power Platform environments, the process of enabling the Microsoft Power Platform integration for the environment can't be automated. The system can't automatically determine to which Power Platform environment the Finance and Operations apps environment should be linked for the Power Platform integration.

**NOTE**

The Microsoft Power Platform integration will automatically be enabled for Finance and Operations apps environments that are already connected to a single Microsoft Power Platform environment and that already use business events functionality. However, these environments won't be able to use the new business events and data events functionality starting in release 10.0.22. The business events endpoints that are already used in these environments will be migrated to the Dataverse platform in version 10.0.23. At that point, the new business events and data events functionality will become available in the environments. Until then, business events will continue to work as they are currently configured in the environment.

For more information about the new business events and data events functionality that will be delayed for these environments until the migration is completed, see Finance and Operations business events in Dataverse and Finance and Operations CUD events in Dataverse in the 2021 release wave 2 plan.
There are two options to enable the Power Platform integration for a Finance and Operations environment that has links to multiple Power Platform environments:

- Reconfigure your dual-write and/or virtual entity solutions to link the Finance and Operations apps environment to the Power Platform environment created at deployment. When all links are configured for the single Power Platform environment, you can enable the Power Platform integration in LCS following the steps outlined in the Enable integration after environment deployment section above. This is the preferred solution because it can be managed without Microsoft support.
- To enable the Power Platform integration with a linked Power Platform environment other than the Power Platform environment created during deployment of the Finance and Operations environment, either work with your FastTrack solution architect or contact Microsoft Support to enable the Power Platform integration with a selected environment.

For more information about dual-write configuration options, see Linking mismatch.

Enable the integration for cloud-hosted development environments

You can manually enable the Microsoft Power Platform integration for cloud-hosted development environments by completing the procedures in this section. For information about how to deploy cloud development environments, see Deploy and access development environments.

Register an application in the Azure portal

<table>
<thead>
<tr>
<th>IMPORTANT</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Azure AD application must be created on the same tenant as the Finance and Operations app.</td>
</tr>
</tbody>
</table>

1. Open the Azure portal.
2. Go to Azure Active Directory > App registrations.
3. Select New registration, and enter the following information:
   - Name – Enter a unique name.
   - Account type – Select Accounts in any organizational directory (Any Azure AD directory - Multitenant).
   - Redirect URI – Leave this field blank.
4. Select Register.
5. Make a note of the Application (client) ID value. You will need this value later.
6. Create a symmetric key for the application.
   a. Select Certificates & secrets in the left navigation pane for the new app registration.
   b. Select New client secret.
   c. Enter a description and an expiration date.
   d. Select Save.
   e. Copy the key in the Value field that is created. You will need this key value later.

Add the Azure AD application as a Microsoft Power Platform user

After the Azure AD application has been created in the Azure portal, it must be added as a Microsoft Power Platform application user.

1. In the Power Platform admin center, create the application user by following the steps in Create an application user.
2. In the step where you select security roles to add for the application user, select **Finance and Operations Integration User**.

**Grant app permissions in Finance and Operations apps**

Dataverse will use the Azure AD application that you created to call Finance and Operations apps. Therefore, the application must be trusted by Finance and Operations apps and associated with a user account that has the appropriate rights.

1. In Finance and Operations apps, go to **System administration > Setup > Azure Active Directory applications**.

2. Select **New** to add a row to the grid, and enter the following information:

   - **Client ID** – Enter the **Application (client) ID** value of the Azure AD application that you created earlier.
   - **Name** – Enter **Dataverse Integration** (or another name that you will recognize for the integration).
   - **User ID** – Select **PowerPlatformApp**.

**NOTE**

The **PowerPlatformApp** user that is available has the appropriate permissions for Dataverse integrations with Finance and Operations apps. However, if this user doesn't exist, or if you want to use a different application user account, you can create or use any other user that has the following roles: **Business events security role**, **Dataverse Virtual entity application**, **Dataverse Virtual entity anonymous user**, and **Dataverse Virtual entity authenticated user**.

**Configure Finance and Operations apps to use the Azure AD application to connect to Dataverse**


2. Copy the following Windows PowerShell script, and save it to the virtual machine (VM) for the Finance and Operations environment as a .ps1 file.

```powershell
param(
    [Parameter(Mandatory = $false)]
    [switch]$Relaunched
)

$isRelaunched = $false
if ($PSBoundParameters.ContainsKey("Relaunched"))
{
    $isRelaunched = $Relaunched.IsPresent
}
{
    # Relaunch as an elevated process:
    Start-Process powershell.exe "-File", (""{0}"" -f $MyInvocation.MyCommand.Path), "-Relaunched" -Verb RunAs
    exit
}

$aosWebsiteName = "AOSService"

function Get-AosWebSitePhysicalPath()
{
    if (Get-Service W3SVC | Where-Object status -ne 'Running')
    {
        #IIS service is not running, starting IIS Service.
        Start-Service W3SVC
    }
```

```powershell
```
```
$webSitePhysicalPath = (Get-Website | Where-Object { $_.Name -eq $aosWebsiteName }).PhysicalPath

return $webSitePhysicalPath

function Set-WebConfigValue($Key, [string]$Value)
{
    $webroot = Get-AosWebSitePhysicalPath
    $webConfigPath = Join-Path $webroot "web.config"
    if (-not (Test-Path $webConfigPath))
    {
        Throw "Unable to find web.config file at '" + $(webConfigPath) + "'..."
    }
    $appSettingNode = $webConfigDocument.SelectSingleNode("/configuration/appSettings/add[@key='$($Key)']")
    if ($null -ne $appSettingNode)
    {
        Write-Host "Updating key '" + $(Key) + "' to value '" + $(Value) + "'..."
        $appSettingNode.Value = [string]$Value
    }
    else
    {
        Write-Host "Inserting new key '" + $(Key) + "' with value '" + $(Value) + "'..."
        $ns = New-Object System.Xml.XmlNamespaceManager($webConfigDocument.NameTable)
        $addElement = $webConfigDocument.CreateElement("add")
        $addElement.SetAttribute("key", $Key)
        $addElement.SetAttribute("value", $Value)
        $appSettings = $webConfigDocument.SelectSingleNode("/ns:appSettings", $ns)
        $appSettings.AppendChild($addElement) | Out-Null
    }
    $webConfigDocument.Save($webConfigPath)
    Write-Host
}

function Confirm-ValueOfType($Value, $Type)
{
    if ($Type -eq "Uri")
    {
        try
        {
            New-Object System.Uri $Value | Out-Null
        }
        catch
        {
            Throw "Cannot parse '" + $(Value) + "' as a URL: " + $( $_ )
        }
    } elseif ($Type -eq "Guid")
    {
        try
        {
            [Guid]::Parse($Value) | Out-Null
        }
        catch
        {
            Throw "Cannot parse '" + $(Value) + "' as a guid: " + $( $_ )
        }
    } elseif ($Type -eq "String")
    {
        if ([string]::IsNullOrEmpty($Value))
        {
            Throw "String value cannot be empty."
        }
    }
function Update-WebConfigValueFromHost($Key, $Prompt, $Type)
{
    $value = Read-Host -Prompt $Prompt
    Confirm-ValueOfType -Value $value -Type $Type
    Set-WebConfigValue -Key $Key -Value $value
}

function Enable-Flight($FlightName)
{
    Write-Verbose "Enabling flight '$($FlightName)'..."
    $webroot = Get-AosWebSitePhysicalPath -ErrorAction stop
    $webrootBinPath = Join-Path $webroot "bin"
    $environmentDllPath = Join-Path $webrootBinPath 'Microsoft.Dynamics.ApplicationPlatform.Environment.dll'
    Add-Type -Path $environmentDllPath
    $ServerName = $config.DataAccess.DbServer
    $DatabaseName = $config.DataAccess.Database
    $UserId = $config.DataAccess.SqlUser
    $Password = $config.DataAccess.SqlPwd
    $EnableFlightQuery = "DECLARE @flightName NVARCHAR(100) = '$($FlightName)';
    IF NOT EXISTS (SELECT TOP 1 1 FROM SysFlighting WHERE flightName = @flightName)
    INSERT INTO SYSFLIGHTING(FLIGHTNAME, ENABLED, FLIGHTSERVICEID, PARTITION)
    SELECT @flightName, 1, 12719367, RECID FROM DBO.[PARTITIONS];
    ELSE
    UPDATE SysFlighting SET enabled = 1, flightServiceId = 12719367 WHERE flightName = @flightName;"
    Invoke-Sqlcmd -ServerInstance $ServerName -Database $DatabaseName -Username $UserId -Password $Password -Query $EnableFlightQuery
    Write-Verbose "Flight '$($FlightName)' has been enabled."
}

function Test-Settings()
{
    $cdsApiPath = "accounts";
    Write-Host "Testing setup by calling API '$($cdsApiPath)'..."
    $webroot = Get-AosWebSitePhysicalPath -ErrorAction stop
    $webrootBinPath = Join-Path $webroot "bin"
    $httpCommunicationDllPath = Join-Path $webrootBinPath "Microsoft.Dynamics.HttpCommunication.dll"
    Add-Type -Path $httpCommunicationDllPath
    $ServerName = $config.DataAccess.DbServer
    $DatabaseName = $config.DataAccess.Database
    $UserId = $config.DataAccess.SqlUser
    $Password = $config.DataAccess.SqlPwd
    $enableFlightQuery = "DECLARE @flightName NVARCHAR(100) = '$($FlightName)';
    IF NOT EXISTS (SELECT TOP 1 1 FROM SysFlighting WHERE flightName = @flightName)
    INSERT INTO SYSFLIGHTING(FLIGHTNAME, ENABLED, FLIGHTSERVICEID, PARTITION)
    SELECT @flightName, 1, 12719367, RECID FROM DBO.[PARTITIONS];
    ELSE
    UPDATE SysFlighting SET enabled = 1, flightServiceId = 12719367 WHERE flightName = @flightName;"
    Invoke-Sqlcmd -ServerInstance $ServerName -Database $DatabaseName -Username $UserId -Password $Password -Query $enableFlightQuery
    Write-Verbose "Flight '$($FlightName)' has been enabled."
}

3. Run the script in Windows PowerShell, and follow the instructions. You will enter the following information:

- **Dataverse Organization URL** – Enter the URL that is used to access Dataverse. For example, enter [https://contoso.crm.dynamics.com](https://contoso.crm.dynamics.com). You can find this URL in the Environment URL field in the Details section of the environment details in the Power Platform admin center.

- **Dataverse Organization ID** – You can find this ID in the Organization ID field in the Details section of the environment details in the Power Platform admin center.

- **Dataverse AAD Tenant domain** – Enter the primary domain of the Azure AD tenant that is used by Dataverse. You can find this domain in the Domain field for the directory on the Portal settings page in the Azure portal. Typically, it’s also the domain segment of the administrator’s email address. For example, if the email address is `admin@contoso.onmicrosoft.com`, the domain is `contoso.onmicrosoft.com`.

- **Dataverse AAD app ID** – Enter the Application (client) ID value of the Azure AD application that you created earlier.

- **Dataverse AAD app secret** – Enter the secret key value that was created earlier for the Azure AD apps.
Troubleshooting the setup

Setup can fail at various stages of the deployment of the Dataverse-based environment.

Any time that the setup fails, an error message is shown. The following illustration shows an example of the error message for a dual-write setup failure.

![Error message for dual-write setup failure](image)

Based on the error message, you might have to address licensing or capacity issues. After these issues have been fixed, you can select **Resume** in the **Power Platform integration** section of the **Environment details** page in LCS to finish the setup.
Finance and Operations apps and Microsoft Power Platform maintain separate user security. Users must have appropriate permissions in each environment to access Finance and Operations apps resources through Microsoft Power Platform. User setup can be simplified by synchronizing users between the two environments.

User provisioning

Creating Finance and Operations apps users in Microsoft Power Platform

There are three ways to automate the creation of Finance and Operations apps users in the Microsoft Power Platform environment:

- When a new Microsoft Power Platform environment is created as a linked environment that is linked to a Finance and Operations apps environment, all active users that are created in the Microsoft 365 admin center and that the Dynamics 365 Finance license is assigned to are automatically added as users in the new Microsoft Power Platform environment. For more information about how to create linked environments, see Enable the Microsoft Power Platform integration.

- When a new user is created in the Microsoft 365 admin center, and the Dynamics 365 Finance license is assigned, the new user is automatically created as a user in Microsoft Power Platform environments that are linked to a Finance and Operations apps environment. The process of creating the new user in Microsoft Power Platform can take up to an hour.

- If a user that the Dynamics 365 Finance license is assigned to signs in to the Microsoft Power Platform environment before user synchronization occurs, the user record is created when the Microsoft Power Platform environment is first accessed.

The creation of Finance and Operations apps users in linked Microsoft Power Platform environments can be automated. However, you can also add Finance and Operations apps users directly in Microsoft Power Platform by following the steps in Add users to an environment that has a Dataverse database.

Creating Microsoft Power Platform users in Finance and Operations apps

Administrators can manually import Microsoft Power Platform users into Finance and Operations apps from Azure Active Directory (Azure AD). To import users from Azure AD, select Import users in Finance and Operations apps. For more information, see Import new users from Azure AD.

Security model

Microsoft Power Platform users who work in Dataverse can interact with Finance and Operations apps entities through virtual entities. They can also receive business events that are triggered by user actions in Finance and Operations apps. To interact with virtual entities in Dataverse, a user requires access to the virtual entity metadata. When a transaction is performed in Dataverse, a virtual entity call is made to Finance and Operations apps. There, the call authorizes the user's request by validating against the user's security role and privileges that are defined in Finance and Operations apps.

For more information about Dataverse virtual entity interaction and the Finance and Operations apps security model, see Architecture.

Security roles in Microsoft Power Platform

When users who have the Dynamics 365 Finance license are automatically created as users in Microsoft Power Platform, both the Finance and Operations Basic User security role and the Environment Maker security role are automatically assigned to the user.
security role are automatically assigned to the Microsoft Power Platform users.

- **Finance and Operations Basic User** – This security role allows the user to access and update virtual entities and business events. When a virtual entity or business event is activated in the Dataverse environment, the virtual entity or business event privileges are automatically granted to the security role.

- **Environment Maker** – This security role allows the user to create apps in Power Apps and flows in Power Automate.

For information about how to manually assign security roles to Microsoft Power Platform users in the Power Platform admin center, see [Assign a security role to a user](#).

### Security roles in Finance and Operations apps

The security roles that are assigned to a user in Finance and Operations apps depend on the system access that the user requires to perform their assigned duties and responsibilities. The security roles in Finance and Operations apps determine the application data that a user has access to. A user can't access data through the Microsoft Power Platform integration if the user's assigned security roles in Finance and Operations apps don't grant permissions to the data.

The security role for Finance and Operations apps must be manually added by the system administrator. Different security roles can be assigned to a user in Finance and Operations apps.
These topics describe dual-write integration.

- What is dual-write?
  - Top reasons to use dual-write
  - What does dual-write mean for developers and architects of customer engagement app?
- What’s new or changed in dual-write
- Frequently asked questions

Dual-write setup

- System requirements for dual-write
- Guidance for how to set up dual-write
- Considerations for initial synchronization
- Dual-write limits for live synchronization
- Dual-write setup from Lifecycle Services
- Enable dual-write for existing Finance and Operations apps
  - Enable dual-write for existing Finance and Operations apps
  - System requirements and prerequisites
  - How to use the dual-write wizard to link your environments
  - Enable table map for dual-write
- Currency data-type migration for dual-write
- Set up the mapping for the sales order status columns
- Filter intercompany orders to avoid synchronizing Orders and OrderLines

Managing dual-write after setup

- Customize table and column mappings
- Customization guidance for dual-write
- Handling multiple table maps
- Edit a legal entity after dual-write setup
- Pause dual-write for maintenance
- Error management and alert notifications
- Application lifecycle management
- User-specified team owner
- Unlink and relink dual-write environments

Mapping concepts between apps

These topics describe mapping between concepts in finance and operations apps and concepts in customer engagement app.
engagement apps.

- Integrated customer master
- Integrated vendor master
  - Switch between vendor designs
- Customer loyalty cards and reward points
- Unified product experience
  - Integrated sites and warehouses
- Company concept in Dataverse
  - Initialize company data
- Organization hierarchy awareness
- Access to finance and tax reference data
  - Integrated ledger
  - Integrated tax master
- Sync on-demand with the Supply Chain Management price engine
- Sync on-demand with the Commerce price engine
- Prospect to cash in dual-write
- Integrate procurement in Supply Chain Management with Field Service
- In-house assets for servicing
- Onhand inventory availability
- Integrated worker, job, and position
- Party and global address book
  - Using Microsoft Power Apps portals with the Party data model
  - Upgrade to the party and global address book model
- Note integration
- Mapping reference

Support

- Support for Field Service and Project Service Automation solutions
- Migrate Prospect to cash data from Data Integrator to dual-write

Troubleshooting

- General troubleshooting
- Troubleshoot issues during initial setup
- Troubleshoot issues during initial synchronization
- Troubleshoot live synchronization issues
- Troubleshoot dual-write issues in Finance and Operations apps
- Troubleshoot party and global address book problems
- Troubleshoot issues related to solution awareness
- Troubleshoot issues from upgrades of Finance and Operations apps
- Verify dual-write configuration in Finance and Operations apps and Dataverse
- Errors codes for table map health check
Dual-write is an out-of-box infrastructure that provides near-real-time interaction between customer engagement apps in Microsoft Dynamics 365 and Finance and Operations apps. To get started with dual-write, see the Dual-write home page.

November 2021 release of party and global address book

The November 2021 release of the Dual-write Party and Global Address Book Solutions 3.3.0.5 contains the following features and bug fixes.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single view for party</td>
<td>We are providing a new form to access Party data. The Party form provides the capability to view and manage party records along with all the associated customers, contacts, and vendors and their postal addresses and electronic addresses from a single form.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>On the Accounts form, postal address updates from the Summary tab do not synchronize, which causes a data mismatch between Microsoft Dataverse and Finance and Operations apps.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>When an electronic address is changed from non-primary to primary, the updates to telephone extension/description fields does not synchronize from msdyn_partyelectronicaddress to Contact table.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Error while updating the Gender field on a Contact record to &quot;non-specific&quot; in Finance and Operations apps.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>In Finance and Operations apps, during address creation, when a second address is marked as primary and saved, the IsPrimary value change does not reflect in Dataverse.</td>
<td>General availability</td>
</tr>
</tbody>
</table>

September 2021 release of party and global address book

The September 2021 hotfix release of the Dual-write Party and Global Address Book Solutions 3.1.0.4 is based on Dual-write core solution version 1.0.29.

This release contains bug fixes listed in the following table.
**FEATURE** | **DESCRIPTION** | **STATUS**
--- | --- | ---
Bug fix | Some client APIs have been deprecated and replaced with new APIs. The JavaScript code has been upgraded to use the new client APIs. | General availability

Bug fix | Portals registration using email address fails when last name is not supplied. | General availability

Bug fix | Unable to create a new postal address for a vendor. | General availability

**Solution details**

<table>
<thead>
<tr>
<th>SOLUTION NAME</th>
<th>HAS NEW CHANGES?</th>
<th>PREVIOUS VERSION</th>
<th>NEW VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Party</td>
<td>Yes</td>
<td>3.1.0.2</td>
<td>3.2.0.4</td>
</tr>
<tr>
<td>Dynamics365GABExtended</td>
<td>Yes</td>
<td>3.1.0.2</td>
<td>3.2.0.4</td>
</tr>
<tr>
<td>Dynamics365GABDualWrite EntityMaps</td>
<td>Yes</td>
<td>3.1.0.2</td>
<td>3.1.0.2</td>
</tr>
<tr>
<td>Dynamics365GABPartyAnchor</td>
<td>Yes</td>
<td>3.1.0.2</td>
<td>3.2.0.4</td>
</tr>
</tbody>
</table>

**August 2021 release**

The August 2021 release of Dual-write application orchestration solution version 2.3.0.15 is based on Dual-write core solution version 1.0.29.

This release contains the features and bug fixes listed in the following table.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fix</td>
<td>Fixes the case where dual-write alerts fail to send.</td>
<td>General availability</td>
</tr>
<tr>
<td>System tables</td>
<td>Adds support for enabling dual-write for system tables.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Some client APIs have been deprecated and replaced with new APIs. The JavaScript code in the dual-write orchestration package has been upgraded to use the new client APIs.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Because dual-write doesn’t support offline mode, the company name does not automatically contain a default value. You must select the company manually.</td>
<td>General availability</td>
</tr>
<tr>
<td>FEATURE</td>
<td>DESCRIPTION</td>
<td>STATUS</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Bug fix</td>
<td>The <strong>Vendor</strong> group field on the <strong>Accounts</strong> form does not filter values based on the selected company.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Saving a <strong>Vendor</strong> record in a Finance and Operations app fails with the error message <strong>Cannot convert the literal '' to the expected type 'Edm.Int32'.</strong></td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>The transformation on the <strong>Vendor payment method</strong> map required an update. The enumeration on the <strong>PAYMENTSTATUS</strong> field is incorrect resulting in error message <strong>Cannot convert the literal 'Confirmed' to expected type 'Edm.Int32'.</strong></td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td><strong>Sales order header</strong> and <strong>Sales order line</strong> maps conflict with <strong>Project contract header</strong> and <strong>Project contract line</strong> maps. You couldn't enable both at once.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Create an error message to state that <strong>Ship To Country/Region</strong> is a mandatory field on <strong>Sales order</strong> and <strong>Purchase order</strong>.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Whenever a sales order is created in Dynamics 365 Sales, the default value of the <strong>Invoice Customer</strong> is based on <strong>Billing Account</strong> value of the <strong>Potential Customer</strong>.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Ability to toggle <strong>Price override</strong> field to true or false.</td>
<td>General availability</td>
</tr>
</tbody>
</table>

This release includes following map changes.

- [CDS sales quotation lines] - [quotedetails] map version 1.0.0.1
- [CDS sales order lines] - [salesorderdetails] map version 1.0.0.1
- [Vendors V2] - [msdyn_vendors] map version 1.0.0.3
- [Vendor payment method] - [msdyn_vendorpaymentmethods] map version 1.0.0.1

**August 2021 release of party and global address book**

The August 2021 release of the Dual-write Party and Global Address Book Solutions 3.1.0.2 is based on Dual-write core solution version 1.0.29.

This release contains features and bug fixes listed in the following table.
### July 2021 release

The July 2021 hotfix release of Dual-write application orchestration solution version is based on Dual-write core solution version 1.0.28.

This release contains the features and bug fixes listed in the following table.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fix</td>
<td>Improved performance when more than 20 legal entities are enabled for dual-write.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Fixed issue with Is primary setting on postal address.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Fill the contact information on Contact for customer or vendor upon party association on main form.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Full name of contact with IsCustomer=Yes or IsVendor=Yes is blank during initial sync and live sync.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Make the country/region field required on both postal address table and customer address table.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Identify a phone number as mobile.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Update the display name from Parties Electronic Addresses table to Party Electronic Addresses table.</td>
<td>General availability</td>
</tr>
<tr>
<td>Party Electronic Address</td>
<td>Synchronize primary electronic address data from lead qualification process, account, and contact creation process to Party Electronic Addresses, and vice versa.</td>
<td>General availability</td>
</tr>
</tbody>
</table>

### June 2021 release

The June 2021 hotfix release ofDual-write application orchestration solution version 2.2.2.98 is based on Dual-write core solution version 1.0.27.

This release contains the features and bug fixes listed in the following table.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracing</td>
<td>Logs basic transaction properties to a remote database for use in analytics and error detection.</td>
<td>General availability</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Status</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>--------</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Implements rollback of transactions that exceed a 2-minute time limit.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Fixes processing of datetime fields during a catch-up sync.</td>
<td>General availability</td>
</tr>
<tr>
<td>Tracing</td>
<td>Limits nonessential plugin trace logs for medium-size transactions (5 or more records).</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Added storage and tracking dimension group lookup fields to the <code>Shared Product Details</code> table. Added these lookup fields to the <code>Released products V2</code> -&gt; <code>msdyn_sharedproductdetails</code> dual-write map.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>In a sales order, when you select an existing product, the <code>Sales product category</code> value defaults to a value that comes from the <code>Product Category Assignments</code> table. But when the defaulted value is not part of sales hierarchy, the line-item addition throws an error message. To prevent the error and allow assignment inside finance and operations apps, the <code>sales product category</code> field can be left blank.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>In a sales order, when you select an existing product, the <code>Sales Product Category</code> is read-only. It is editable when the product is set to write-in.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>The <code>Total Tax</code> field in a sales quotation and sales order are read-only. When the <code>Price per unit</code> field is changed in the sales quotation line or sales order line in finance and operations apps, it must sync back to the respective sales quotation or sales order line in customer engagement apps.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Updating the <code>Warehouse name</code> field in finance and operations apps caused the <code>Name</code> field in Dataverse to be blanked out.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Localization fixes for Ukraine.</td>
<td>General availability</td>
</tr>
</tbody>
</table>

**May 2021 release**

The May 2021 hotfix release of Dual-write application orchestration solution version 2.2.2.60 is based on Dual-write core solution version 1.0.26.
This release contains the features and bug fixes listed in the following table.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Government Community Cloud support</td>
<td>Dual-write runtime on the Government Community Cloud region is supported.</td>
<td>General availability</td>
</tr>
<tr>
<td>User-friendly error messages</td>
<td>Enables user-friendly error messages for some of the live sync failures.</td>
<td>General availability</td>
</tr>
</tbody>
</table>

**May 2021 release of party and global address book**

The May 2021 hotfix release of the Dual-write Party and Global Address Book Solutions 3.0.0.26 is based on Dual-write core solution version 1.0.24.

This release contains the bug fixes listed in the following table.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fix</td>
<td>In customer engagement apps, when a lead contains postal address and is qualified, the postal address is associated with the account. It doesn't flow to Finance and Operations apps, however.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>In customer engagement apps, when you add an address to an existing account or contact, the address doesn't flow to Finance and Operations apps.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>More address fields are added to the Customer Address table.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>In Dataverse, we changed the display name of msdyn_contactforparties table to <strong>Contact for Customer or Vendor</strong></td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>We fixed a language transformation in the Contacts V2 (msdyn_contactforparties) mapping.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>We fixed an issue in initial sync in the CDS Party postal address locations (msdyn_partypostaladdresses) mapping to avoid loss of some records in Customer Address table.</td>
<td>General availability</td>
</tr>
</tbody>
</table>

**Solution details**

<table>
<thead>
<tr>
<th>SOLUTION NAME</th>
<th>HAS NEW CHANGES?</th>
<th>PREVIOUS VERSION</th>
<th>NEW VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Party</td>
<td>Yes</td>
<td>3.0.0.1</td>
<td>3.0.0.26</td>
</tr>
<tr>
<td>SOLUTION NAME</td>
<td>HAS NEW CHANGES?</td>
<td>PREVIOUS VERSION</td>
<td>NEW VERSION</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>------------------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Dynamics365GABExtended</td>
<td>Yes</td>
<td>3.0.0.1</td>
<td>3.0.0.26</td>
</tr>
<tr>
<td>Dynamics365GABDualWriteEntityMaps</td>
<td>Yes</td>
<td>3.0.0.1</td>
<td>3.0.0.26</td>
</tr>
<tr>
<td>Dynamics365GABPartyAnchor</td>
<td>Yes</td>
<td>3.0.0.1</td>
<td>3.0.0.26</td>
</tr>
<tr>
<td>Dynamics365GABPartyCommon</td>
<td>Yes</td>
<td>3.0.0.1</td>
<td>3.0.0.26</td>
</tr>
</tbody>
</table>

**Map instructions**

Follow these steps to apply the new maps:

1. Apply the latest map version 1.0.0.2 for CDS Postal address history V2 (msdyn_postaladdresses) mapping.
2. Apply the latest map version for Contacts V2 (msdyn_contactforparties) mapping.
3. Run the initial sync of the CDS Party postal address locations (msdyn_partypostaladdresses) mapping twice to make sure that there is no loss of address records in Customer Address table due to concurrent address updates.

**April 2021 release**

The April 2021 hotfix release of the Dual-write application orchestration solution version 2.2.2.60 is based on Dual-write core solution version 1.0.25.

This release contains the features and bug fixes listed in the following table.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record deletion</td>
<td>Handle record deletion during transactions with multiple entities.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Handle conflict resolution during catch-up sync.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Issues related to solution import on environments.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>The Notes table now understands the Null value.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Dual write orchestration package 2.2.2.50 does not replace the existing key (msdyn_locationid field) on the Address table with the new key which is a combination of the msdyn_locationid and parentid fields. Instead it shows both keys. This has been fixed with the new version 2.2.2.60. This new version is applicable only when you are using the party and global address book solution.</td>
<td>General availability</td>
</tr>
</tbody>
</table>
March 2021 release

The March 2021 release of the Dual-write application orchestration solution version 2.2.2.50 is based on Dual-write core solution version 1.0.24.

This release contains the features and bug fixes listed in the following table.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Party and Global Address Book</td>
<td>Brings schema parity with Finance and Operations apps, and gives you the ability to solve complex business problems on Dataverse related to customers, vendors, and contact persons. To use this feature, install Dual-write Party and Global Address Book Solutions. Supported Finance and Operations versions are 10.0.605.30025 (platform update 14), 10.0.644.20031 (platform update 15), 10.0.689.10027 (platform update 16), and 10.0.761.1 (platform update 17)</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Product category name is part of the natural/integration key of the Product Categories table. Updating the name using a Finance and Operations app cause an insert in Dataverse instead of an update. Please use the new map for msdyn_productcategories - Product categories with version 1.0.0.1. The supported Finance and Operations version is 10.0.778.0 (platform update 42)</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Localization bug fixes and updates.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>A note without a description throws error.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>In Finance and Operations apps, running the “Calculate Sales Totals” batch job updates all orders modified within last 24 hours and fixes the totals regardless of the status of the order for example, canceled or fulfilled. That action triggers a re-cancellation or re-fulfillment causing a conflict error.</td>
<td>General availability</td>
</tr>
</tbody>
</table>

February 2021 release

The February 2021 release of the Dual-write application orchestration solution version 2.2.2.23 is based on Dual-write core solution version 1.0.24 and version 10.0.16 (10.0.689.10004) or newer of Finance and Operations apps and version 9.1.0000.11732 or newer of Dataverse.

This release contains the features and bug fixes listed in the following table.
<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commerce price engine for sales quotations</td>
<td>Get pricing for sales quotations using commerce price engine.</td>
<td>General availability</td>
</tr>
<tr>
<td>Notes integration</td>
<td>Notes are integrated between customer engagement apps and Finance and Operations applications for customers, vendors, sales orders, and purchase orders.</td>
<td>General availability</td>
</tr>
</tbody>
</table>

**IMPORTANT**

If you don't need notes integration, do not install or upgrade to Dual-write application orchestration solution version 2.2.2.23 or later. If you install the update, you won't be able to uninstall the notes feature.

**January 2021 release**

The January 2021 release of the Dual-write application orchestration solution version 2.2.1.30 is based on Dual-write core solution version 1.0.24 and version 10.0.14 of Finance and Operations apps.

This release contains the features and bug fixes listed in the following table.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fix</td>
<td>French-localized strings in the user interface exceeded the maximum limit of 100 characters.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Error while starting the Dataverse released distinct products map.</td>
<td>General availability</td>
</tr>
</tbody>
</table>

The January 2021 release of the Dual-write application orchestration solution version 2.2.1.23 is based on Dual-write core solution version 1.0.24 and version 10.0.14 of Finance and Operations apps.

This release contains the features and bug fixes listed in the following table.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purchase order integration</td>
<td>Integrates purchase order functionality between Dynamics 365 Field Service and Dynamics 365 Supply Chain Management.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Localization updates.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>In customer engagement apps, on the Contact form, after you set Is Sellable to Yes and save the record, the contact is considered a customer who can transact. Because customers are associated with transactions, Is Sellable becomes read-only after saving. You can't change it back to No.</td>
<td>General availability</td>
</tr>
</tbody>
</table>
December 2020 release

The December 2020 release of the Dual-write core solution (1.0.24) contains the features and bug fixes listed in the following table.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal repeater service</td>
<td>Enables the dual-write runtime plugin to communicate with the Finance and Operations signal repeater service with authentication support.</td>
<td>General availability</td>
</tr>
</tbody>
</table>

November 2020 release

The November 2020 release of the Dual-write core solution (1.0.23) contains the features and bug fixes listed in the following table.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication</td>
<td>Support for new authentication certificate to ensure security.</td>
<td>General availability</td>
</tr>
</tbody>
</table>

October 2020 release

The October 2020 release of the Dual-write application orchestration solution and the Dual-write core solution contains the features and bug fixes listed in the following table.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camel-cased column mappings</td>
<td>Adds support for column mappings with camel-cased navigation properties.</td>
<td>General availability</td>
</tr>
</tbody>
</table>

Bug fix

Fixes the bug where an unrecognized tag configuration would cause dual-write execution to be skipped

General availability

September 2020 release

The September 2020 release of the Dual-write application orchestration solution version 2.0.777.493 is based on Dual-write core solution version 1.0.21.

The September 2020 release contains the features and bug fixes listed in the following table.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Lead qualification process in Sales is now company striped

Dynamics 365 Sales users can create a lead, qualify the lead to an opportunity, convert an opportunity into a quote, activate a quote, and create an order. This process was broken in dual-write due to lack of company striping on the Lead entity. We implemented company striping on the Lead entity, which cascades the company to the underlying Account and Opportunity tables. Thus the application behavior is restored to support the process. During the Lead qualification process, the Contact entity isn't company striped. This design supports the Party entity model that is due in October 2020. To learn about the Party and GlobalAddressBook model for dual-write, join the dual-write Yammer group.

### Map state transitions from Order to SalesOrder

The Order form in Dynamics 365 Sales is always set to Active. To create state transitions from Order in Dynamics 365 Sales to SalesOrder in Dynamics 365 Supply Chain Management, we introduced the ProcessingStatus column.

### Money to decimal data type conversion

Dataverse environments are limited to 4 decimal places for currency and 10 decimal places for exchange rates. Finance and Operations apps support more decimal places than Dataverse. You can now opt in to extend the decimal support in Dataverse to help ensure there’s no loss of decimal place data when using dual-write.

### Security role for company and currency exchange

Company and currency exchange tables are global in nature and all dual-write users require read access to these 2 tables. To simplify the experience, we’ve added a new security role named dual-write app user. Each dual-write user must be added to this security role.

### Security role for setup

Adds the Dual-write Runtime User security role. This role allows non-administrator users to create rows that are set up for dual-write. This feature is part of Dual-write core solution 10.0.21.

### Tracing

Internal column added for use in tracing. This feature is part of Dual-write core solution 10.0.21.
<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fix</td>
<td>Fixes issues where dual-write fails because of a mismatch between the plugin and the destination environments. This fix is part of Dual-write core solution 10.0.21.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Support to ensure that unused plugins are deleted. This fix is part of Dual-write core solution 10.0.21.</td>
<td>General availability</td>
</tr>
</tbody>
</table>

### August 2020 release

The August 2020 release of the dual-write orchestration package contains the features and bug fixes listed in the following table.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manage multiple table maps</td>
<td>As part of day-to-day operations, you might need to bulk handle table maps. For example, you might want to simultaneously enable or pause a set of table maps. Instead of doing this one-by-one, which is cumbersome and time consuming, you can now enable, pause, resume, or stop more than one table map at the same time in the dual-write list page.</td>
<td>General availability</td>
</tr>
<tr>
<td>Bug fix</td>
<td>Fixes issues where rows would be skipped in certain cases during project execution. This fix is part of Dual-write core solution version 10.0.19.</td>
<td>General availability</td>
</tr>
</tbody>
</table>

### June 2020 release

The June 2020 release of the dual-write orchestration package contains the features and bug fixes listed in the following table.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit legal entity after setup</td>
<td>The company or legal entity list isn’t static and is constantly changing. You might need to add new companies, for example, during a phased rollout or acquisition. Previously, you couldn’t add a company or legal entity without system downtime. During this downtime, you would have to unlink and relink your environment. That can be expensive, especially if you have pre-existing data. With this feature, you can add a company in a live environment without having to unlink and relink.</td>
<td>General availability</td>
</tr>
</tbody>
</table>
May 2020 release

The May 2020 release of the dual-write orchestration package (version 2.0.777.353) contains the features and bug fixes listed in the following table.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look up on-hand inventory</td>
<td>Ability to look up on-hand inventory and available-to-promise dates on forms in customer engagement apps.</td>
<td>General availability</td>
</tr>
<tr>
<td>Unit conversions</td>
<td>When unit conversions occur in a Finance and Operations app at the quote line and order line, the customer engagement app honors the unit conversion and reflects the respective changes to unit and price in the customer engagement app quote detail and order detail.</td>
<td>General availability</td>
</tr>
<tr>
<td>Currency change restriction</td>
<td>When you try to change the currency in a Finance and Operations app for an existing quote or order, the change fails.</td>
<td>General availability</td>
</tr>
<tr>
<td>Parity in <strong>Account</strong> and <strong>Contact</strong> forms</td>
<td>Bring attribute parity in <strong>Account</strong> and <strong>Contact</strong> forms in customer engagement apps for B2B and B2C customers.</td>
<td>General availability</td>
</tr>
<tr>
<td>No address duplication</td>
<td>Don’t duplicate an address in a Finance and Operations app when there’s a create or update action on a customer engagement app quote or order.</td>
<td>General availability</td>
</tr>
<tr>
<td><strong>SalesTaxGroup</strong> support</td>
<td>Support for <strong>SalesTaxGroup</strong> in <strong>Account</strong> and <strong>Contact</strong> forms for business-to-business (B2B) and business-to-consumer (B2C) customers.</td>
<td>General availability</td>
</tr>
<tr>
<td>Create sellable contacts</td>
<td>Allow creation of a sellable contact using the <strong>Quick Create: Contact</strong> form in customer engagement apps.</td>
<td>General availability</td>
</tr>
<tr>
<td>Quote and order creation</td>
<td>Enable quote and order creation for B2C customers.</td>
<td>General availability</td>
</tr>
<tr>
<td>Removal of tenant admin-level consent requirement</td>
<td>Until now, before you could enable dual-write, a tenant admin needed to explicitly give consent to the applications. This wasn’t always practical and required additional approval, which can be time consuming. With this feature, we removed this prerequisite and the need for explicitly giving consent to the applications.</td>
<td>General availability</td>
</tr>
</tbody>
</table>
### Force unlink dual-write environment

Previously, while testing dual-write, you had to disable all the table maps before unlinking a dual-write environment. This seemed cumbersome and sometimes not possible if one of the environments wasn't available. This new feature provides a quick way to unlink your test and trial environments.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Force unlink dual-write environment</td>
<td>Previously, while testing dual-write, you had to disable all the table maps before unlinking a dual-write environment. This seemed cumbersome and sometimes not possible if one of the environments wasn't available. This new feature provides a quick way to unlink your test and trial environments.</td>
<td>General availability</td>
</tr>
</tbody>
</table>
What is dual-write?

Dual-write is an out-of-box infrastructure that provides near-real-time interaction between customer engagement apps and Finance and Operations apps. When data about customers, products, people, and operations flows beyond application boundaries, all departments in an organization are empowered.

Dual-write provides tightly coupled, bidirectional integration between Finance and Operations apps and Dataverse. Any data change in Finance and Operations apps causes writes to Dataverse, and any data change in Dataverse causes writes to Finance and Operations apps. This automated data flow provides an integrated user experience across the apps.

Dual-write has two aspects: an infrastructure aspect and an application aspect.

**Infrastructure**

The dual-write infrastructure is extensible and reliable, and includes the following key features:

- Synchronous and bidirectional data flow between applications
- Synchronization, together with play, pause, and catchup modes to support the system during online and offline/asynchronous modes.
- Ability to sync initial data between the applications
- Combined view of activity and error logs for data admins
- Ability to configure custom alerts and thresholds, and to subscribe to notifications
- Intuitive user interface (UI) for filtering and transformations
- Ability to set and view table dependencies and relationships
- Extensibility for both standard and custom tables and maps
- Reliable application lifecycle management
- Out-of-box setup experience for new customers

**Application**

Dual-write creates a mapping between concepts in Finance and Operations apps and concepts in customer engagement apps. This integration supports the following scenarios:

- Integrated customer master
- Access to customer loyalty cards and reward points
- Unified product mastering experience
- Awareness of organization hierarchy
- Integrated vendor master
- Access to finance and tax reference data
- On-demand price engine experience
- Integrated prospect-to-cash experience
- Ability to serve both in-house assets and customer assets through field agents
- Integrated procure-to-pay experience
- Integrated activities and notes for customer data and documents
- Ability to look up on-hand inventory availability and details
- Project-to-cash experience
- Ability to handle multiple addresses and roles through the party concept

Top reasons to use dual-write

Dual-write provides data integration across Microsoft Dynamics 365 applications. This robust framework links environments and enables different business applications to work together. Here are the top reasons why you should use dual-write:

- Dual-write provides tightly coupled, near-real-time, and bidirectional integration between finance and operations apps and customer engagement apps. This integration makes Microsoft Dynamics 365 the one-stop shop for all your business solutions. Customers who use Dynamics 365 Finance and Dynamics 365 Supply Chain Management, but who use non-Microsoft solutions for customer relationship management (CRM), are moving toward Dynamics 365 for its dual-write support.
- Data from customers, products, operations, projects, and the Internet of Things (IoT) automatically flows to Dataverse through dual-write. This connection is useful for businesses that are interested in Power Platform expansions.
- The dual-write infrastructure follows the no-code/low-code principle. Minimal engineering effort is required to extend the standard table-to-table maps and to include custom maps.
- Dual-write supports both online mode and offline mode. Microsoft is the only company that offers support for online and offline modes.

What does dual-write mean for developers and architects of customer engagement apps?

Dual-write automates the data flow between Finance and Operations apps and customer engagement apps. Dual-write consists of two AppSource solutions that are installed on Dataverse. The solutions expand the table schema, plugins, and workflows on Dataverse so that they can scale to ERP size. For a successful implementation, developers and architects of customer engagement apps must understand these changes and collaborate with their counterparts on Finance and Operations apps.

To create parity with Finance and Operations applications, dual-write makes some crucial changes in the Dataverse schema. If you understand the plan, you can avoid some design and development rework in the future.

- When the dual-write AppSource package is installed, Dataverse will have new concepts such as company and party. These concepts help applications built on Dataverse, including Dynamics 365 Sales, Dynamics 365 Marketing, Dynamics 365 Customer Service, and Dynamics 365 Field Service, to interact seamlessly with Finance and Operations apps.
- Activities and notes are unified and expanded to support both C1s (users of the system) and C2s (customers of the system).
- To prevent data loss during currency transmission between Finance and Operations apps and the
Dataverse, you’ll be able to extend the number of decimal places in the currency data type of customers engagement apps. The feature autotranslates existing rows to the new extended state at the metadata layer. During this process, the currency value is translated to decimal data rather than money data, and the currency value supports 10 decimal places. This feature is opt-in, and organizations that don’t need more than 4 decimal places of precision do not need to opt in. For more information, see Currency data-type migration for dual-write.

- **Date effectivity** will be added to Dataverse. It will support past, present, and future data on the same table.
- **Product unit conversions** are supported for products, quotes, orders, and invoices.

For more information about upcoming changes, see What’s new or changed in dual-write.
The setup of a dual-write connection has the following requirements:

- Finance and Operations apps that have build version 10.0.9 (10.0.383.20013) (Quality update) and platform update 33 or later
- Customer engagement apps that have platform version 9.1.0000.11732 or later

Dual-write has these limitations:

- You can't run dual-write and the Prospect to cash solution for Data integrator side by side. If you're running the Prospect to cash solution for Data integrator, you must uninstall it.
- Dual-write setup is not supported on trial instances of Finance and Operations apps.
- Dual-write must be used to integrate a single Finance and Operations app instance and a single customer engagement app instance.
- Dual-write currently has an initial synchronization limit of 40 legal entities.
- Dual-write live synchronization has a limit of 250 legal entities.
- Dual-write does not support cross-company data sharing.
- Dual-write requires that the Finance and Operations app and the customer engagement app must be in the same Microsoft Azure Active Directory (Azure AD) tenant.
- Dual-write requires that the Finance and Operations app and the customer engagement app must be deployed in the same Microsoft Azure datacenter.
- Dual-write is not triggered by the doInsert, doUpdate, and doDelete events of Finance and Operations apps. Use the Insert, Update, and Delete events in Finance and Operations apps when you want to trigger dual-write.
- Dual-write doesn't support distributed transactions. For example, if the product receipt posting process is cancelled, dual-write might create the product receipt in Dataverse but not create it in Supply Chain Management.

One Version

Future updates of the dual-write solution will be available through One Version.
You can set up a dual-write connection between a Finance and Operations environment and a Dataverse environment.

- A **Finance and Operations environment** provides the underlying platform for **Finance and Operations apps** (for example, Microsoft Dynamics 365 Finance, Dynamics 365 Supply Chain Management, Dynamics 365 Commerce, and Dynamics 365 Human Resources).
- A **Dataverse environment** provides the underlying platform for **customer engagement apps** (Dynamics 365 Sales, Dynamics 365 Customer Service, Dynamics 365 column Service, Dynamics 365 Marketing, and Dynamics 365 Project Service Automation).

The setup mechanism varies, depending on your subscription and the environment:

- For new instances of Finance and Operations apps, the setup of a dual-write connection begins in Microsoft Dynamics Lifecycle Services (LCS). If you have a license for Microsoft Power Platform, you will get a new Dataverse environment if your tenant doesn’t have one.
- For existing instances Finance and Operations apps, the setup of a dual-write connection begins in the Finance and Operations environment.

Before you start dual-write on an entity, you can run an initial synchronization to handle existing data on both sides: Finance and Operations apps and customer engagement apps. You can skip the initial synchronization if you don’t have to sync data between the two environments.

An initial synchronization lets you copy existing data from one app to another bidirectionally. There are several setup scenarios, depending on the environments that you already have and the type of data in them.

The following setup scenarios are supported:

- A new Finance and Operations app instance and a new customer engagement app instance
- A new Finance and Operations app instance and an existing customer engagement app instance
- A new Finance and Operations app instance that has data and a new customer engagement app instance
- A new Finance and Operations app instance that has data and an existing customer engagement app instance
- An existing Finance and Operations app instance and a new customer engagement app instance
- An existing Finance and Operations app instance and an existing customer engagement app instance

A new Finance and Operations app instance and a new customer engagement app instance
To set up a dual-write connection between a new instance of a Finance and Operations app that has no data and a new instance of a customer engagement app, follow the steps in Dual-write setup from Lifecycle Services. When the connection setup is completed, the following actions automatically occur:

- A new, empty Finance and Operations environment is provisioned.
- A new, empty instance of a customer engagement app is provisioned, where the CRM prime solution is installed.
- A dual-write connection is established for DAT company data.
- Table maps are enabled for live synchronization.

Both environments are then ready for live data synchronization.

A new Finance and Operations app instance and an existing customer engagement app instance

To set up a dual-write connection between a new instance of a Finance and Operations app that has no data and an existing instance of a customer engagement app, follow the steps in Dual-write setup from Lifecycle Services. When the connection setup is completed, the following actions automatically occur:

- A new, empty Finance and Operations environment is provisioned.
- A dual-write connection is established for DAT company data.
- Table maps are enabled for live synchronization.

Both environments are then ready for live data synchronization.

To sync the existing Dataverse data to the Finance and Operations app, follow these steps.

1. Create a new company in the Finance and Operations app.
2. Add the company to the dual-write connection setup.
4. Run the Initial sync functionality for the tables that you want to sync data for.

For links to an example and an alternative approach, see the Example section later in this topic.

A new Finance and Operations app instance that has data and a new customer engagement app instance

To set up a dual-write connection between a new instance of a Finance and Operations app that has data and a new instance of a customer engagement app, follow the steps in the A new Finance and Operations app instance and a new customer engagement app instance section earlier in this topic. When the connection setup is completed, if you want to sync the data to the customer engagement app, follow these steps.

1. Open the Finance and Operations app from the LCS page, sign in, and then go to Data Management > Dual-write.
2. Run the Initial sync functionality for the tables that you want to sync data for.

For links to an example and an alternative approach, see the Example section.
To set up a dual-write connection between a new instance of a Finance and Operations app that has data and an existing instance of a customer engagement app, follow the steps in the **A new Finance and Operations app instance and an existing customer engagement app instance** section earlier in this topic. When the connection setup is completed, if you want to sync the data to the customer engagement app, follow these steps.

1. Open the Finance and Operations app from the LCS page, sign in, and then go to **Data Management > Dual-write**.
2. Run the **Initial sync** functionality for the tables that you want to sync data for.

To sync the existing Dataverse data to the Finance and Operations app, follow these steps.

1. Create a new company in the Finance and Operations app.
2. Add the company to the dual-write connection setup.
3. **Bootstrap** the Dataverse data by using a three-letter ISO company code.
4. Run the **Initial sync** functionality for the tables that you want to sync data for.

For links to an example and an alternative approach, see the **Example** section.

**An existing Finance and Operations app instance and a new customer engagement app instance**

The setup of a dual-write connection between an existing instance of a Finance and Operations app and a new instance of a customer engagement app occurs in the Finance and Operation environment.

1. **Set up the connection from the Finance and Operations app.**
2. Run the **Initial sync** functionality for the tables that you want to sync data for.

For links to an example and an alternative approach, see the **Example** section.

**An existing Finance and Operations app instance and an existing customer engagement app instance**

The setup of a dual-write connection between an existing instance of a Finance and Operations app and an existing instance of a customer engagement app occurs in the Finance and Operation environment.

1. **Set up the connection from the Finance and Operations app.**
2. To sync the existing Dataverse data to the Finance and Operations app, **bootstrapping** the Dataverse data by using a three-letter ISO company code.
3. Run the **Initial sync** functionality for the tables that you want to sync data for.

For links to an example and an alternative approach, see the **Example** section.

**Example**

For an example, see **Enabling the Customers V3—Contacts table map**

For an alternative approach that is based on data volumes in each entity that must run an initial synchronization, see **Considerations for initial synchronization**.
Before you start dual-write on a table, you can run an initial synchronization to handle existing data on both sides: Finance and Operations apps and customer engagement apps. You can skip the initial synchronization if you don’t have to sync data between the two environments.

The initial synchronization lets you copy existing data from one app to another bidirectionally, and there are several considerations when you run it. For example, you might have to migrate data before your go-live. In this case, data can be loaded into one side through data migration and then synced to the other side through the initial synchronization.

We recommend that you use the following approach for the initial synchronization:

- **Single-threaded tables**: First migrate data into the Finance and Operations app, and then trigger the initial synchronization to move the data over to Dataverse. Based on lab testing that Microsoft has done, this sequence has better performance than synchronization from Dataverse to Finance and Operations apps.
- **Multi-threaded tables**: First migrate data into Dataverse, and then trigger the initial synchronization to move the data over to the Finance and Operations app.

**Constraints**

**Data migration slow-down with enabled dual-write**

If you first activate the map in dual-write and then start to import data, migration performance will be poor. We recommend that you not activate running maps in dual-write until the data migration is completed.

**Limit of 500,000 rows per run**

The maximum number of rows that is allowed through initial synchronization is 500,000 per run. The limit of 500,000 rows applies to each legal table, because each legal entity runs separately. For more information, see [*Integrate data into Dataverse*](#). In particular, pay attention to the note that states, “To optimize performance and not overload the apps, we currently limit project executions to 500k rows per execution per project.”

If there must be more than 500,000 rows in a run when you the initial synchronization, we recommend that you migrate data into the Finance and Operations app and Dataverse separately, and skip the initial synchronization.

**Twenty-four-hour limit**

If you're running the initial synchronization from Dataverse to the Finance and Operations app, the import result must be received back from the Finance and Operations app within 24 hours. Otherwise, a time-out occurs. Therefore, if you're syncing lots of data, and the single run takes more than 24 hours, the initial synchronization might fail because of a time-out. For example, an initial synchronization from Dataverse to a Finance and Operations app for the **Customer/Account** table involves 70,000 rows. Therefore, the run might take more than 24 hours and time out.

Don’t run the initial synchronization from Dataverse to a Finance and Operations app for **single-threaded tables** if the data volume is more than 70,000 rows. Because these tables don’t support multi-threading during import, a time-out might occur if the volume is more 70,000 rows. In this situation, you should migrate data into the Finance and Operations app and Dataverse separately, and skip the initial synchronization.

**Limit of 40 legal entities while the environments are being linked**

Currently, there is a limit of 40 legal entities while the environments are being linked. If you try to enable maps where more than 40 legal entities are linked between the environments, you will receive the following error.
Initial synchronization isn't currently supported for table maps that have 10 or more lookups

This limitation applies only to the initial synchronization from Dataverse for table maps that have 10 or more lookups. If you run the initial synchronization against a table map that has 10 or more lookups, you might receive the following error message:

5 Attempts to get data from https://XXXX.azure-apim.net/apim... Failed

As a workaround you can split the initial sync into these steps:

1. Remove some of the lookup columns that are not mandatory from the dual-write table map and bring the number of lookups to 10.
2. After the lookup columns are removed, save the map and do the initial sync.
3. After the initial sync for the first step is successful, add the remaining lookup columns and remove the lookup columns that were synced in first step. Once again make sure the number of lookup columns is 10. Save the map and run the initial sync. Repeat these steps to make sure all the lookup columns are synced.
4. Add all the lookup columns back to the map, save the map and run the map with skip initial sync. This will enable the map for live sync mode.

Five-minute limit for Finance and Operations data export

If you're running the initial synchronization from the Finance and Operations app to Dataverse and the Finance and Operations data export takes more than five minutes, then the initial sync might time out. The time-out can happen if the data table has virtual columns with the `postLoad` method, or the export query isn't optimized (for example, if it has missing indexes).

This type of synchronization is supported in Platform update 37 (PU37) and later. Therefore, you should update your Finance and Operations app to PU37 or later.

Security role for write access

Every user in a customer engagement organization with dual-write must be added to the Dual-Write Runtime User role. Without this role, users will be unable to create any rows in tables in the customer engagement organization.

Company and Currency Exchange Tables Required Security Role

Company and currency exchange tables are global in nature and all dual-write users require read access to these 2 tables. To provide access, all dual-write users will need to be added to the Dual-Write App User security role. If a user does not have this security role assigned to them, they will be unable to read tables that contain Company and Currency values.

Error handling capabilities

Initial synchronization is always a full push

If an individual row fails to be synced, you can't resync only that individual row. The initial synchronization always pushes the whole data set. This behavior is known as a full push. If the initial synchronization only partially succeeds, a second synchronization runs for all the rows, not just the rows that failed to be synced during the initial synchronization.

Only the top five errors can be viewed

You can view only the top five errors from the initial synchronization error log.
Known issues

For information about known issues, see Troubleshoot issues during initial synchronization.

### Guidance matrix

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APP INSTANCE</th>
<th>DATaverse INSTANCE</th>
<th>HAS DATA TO RUN THE INITIAL SYNCHRONIZATION</th>
<th>DESCRIPTION</th>
<th>MAXIMUM VOLUME IN A TABLE</th>
<th>SINGLE-THREADED OR MULTI-THREADED</th>
<th>APPROACH</th>
</tr>
</thead>
<tbody>
<tr>
<td>New</td>
<td>New</td>
<td>No</td>
<td>A new Finance and Operations app instance and a new customer engagement app instance, where neither app has initial data</td>
<td>Not applicable</td>
<td>Any</td>
<td>• Activate dual-write, and skip the initial synchronization.</td>
</tr>
<tr>
<td>New</td>
<td>New</td>
<td>Yes</td>
<td>A new Finance and Operations app instance and a new customer engagement app instance, where one of the apps has migrated data</td>
<td>&lt; 500,000</td>
<td>Single-threaded</td>
<td>1. Migrate data to the Finance and Operations app.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2. Run the initial synchronization.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1. Migrate data to Dataverse.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2. Run the initial synchronization.</td>
</tr>
</tbody>
</table>
| New | Existing | Yes | A new Finance and Operations app instance and an existing customer engagement app instance | < 70,000 | Single-threaded | 1. Create a new company in the Finance and Operations app.  
2. Bootstrap Dataverse for the company code.  
3. Run the initial synchronization. |
<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APP INSTANCE</th>
<th>DATaverse INSTANCE</th>
<th>HAS DATA TO RUN THE INITIAL SYNCHRONIZATION</th>
<th>MAXIMUM VOLUME IN A TABLE</th>
<th>SINGLE-THREADED OR MULTI-THREADED APPROACH</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 70,000</td>
<td>Single-threaded</td>
<td></td>
<td></td>
<td>1. Create a new company in the Finance and Operations app.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2. Bootstrap Dataverse for the company code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3. Migrate data to each app outside the initial synchronization.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4. Activate dual-write, and skip the initial synchronization.</td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS APP INSTANCE</td>
<td>DATaverse INSTANCE</td>
<td>HAS DATA TO RUN THE INITIAL SYNCHRONIZATION</td>
<td>DESCRIPTION</td>
<td>MAXIMUM VOLUME IN A TABLE</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>--------------------</td>
<td>-----------------------------------------------</td>
<td>-------------</td>
<td>--------------------------</td>
</tr>
</tbody>
</table>
|                                    |                    |                                               | < 500,000   | Multi-threaded           |                                   | 1. Create a new company in the Finance and Operations app.  
2. Bootstrap Dataverse for the company code.  
3. Run the initial synchronization. |
<table>
<thead>
<tr>
<th>Finance and Operations App Instance</th>
<th>DataVerse Instance</th>
<th>Has Data to Run the Initial Synchronization</th>
<th>Maximum Volume in a Table</th>
<th>Single-threaded or Multi-threaded</th>
<th>Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existing</td>
<td>New</td>
<td>Yes</td>
<td>&lt; 500,000</td>
<td>Any</td>
<td>Run the initial synchronization.</td>
</tr>
</tbody>
</table>

1. Create a new company in the Finance and Operations app.
2. Bootstrap Dataverse for the company code.
3. Migrate data to each app outside the initial synchronization.
4. Activate dual-write, and skip the initial synchronization.
<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APP INSTANCE</th>
<th>DATaverse INSTANCE</th>
<th>HAS DATA TO RUN THE INITIAL SYNCHRONIZATION</th>
<th>DESCRIPTION</th>
<th>MAXIMUM VOLUME IN A TABLE</th>
<th>SINGLE-THREADED OR MULTI-THREADED</th>
<th>APPROACH</th>
</tr>
</thead>
</table>
| Existing                               | Existing            | Yes                                           | An existing Finance and Operations app instance and an existing customer engagement app instance | < 70,000     | Single-threaded            | 1. Bootstrap Dataverse for the company code.  
2. Run the initial synchronization. |

1. Migrate data to each app.  
2. Activate dual-write, and skip the initial synchronization.
<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APP INSTANCE</th>
<th>DATaverse INSTANCE</th>
<th>HAS DATA TO RUN THE INITIAL SYNCHRONIZATION</th>
<th>MAXIMUM VOLUME IN A TABLE</th>
<th>SINGLE-THREADED OR MULTI-THREADED</th>
<th>APPROACH</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 70,000</td>
<td></td>
<td>Single-threaded</td>
<td></td>
<td>1. Bootstrap Dataverse for the company code.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2. Migrate data to each app outside the initial synchronization.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3. Activate dual-write, and skip the initial synchronization.</td>
<td></td>
</tr>
<tr>
<td>&lt; 500,000</td>
<td></td>
<td>Multi-threaded</td>
<td></td>
<td>1. Bootstrap Dataverse for the company code.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2. Run the initial synchronization.</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS APP INSTANCE</td>
<td>DATaverse INSTANCE</td>
<td>HAS DATA TO RUN THE INITIAL SYNCHRONIZATION</td>
<td>DESCRIPTION</td>
<td>MAXIMUM VOLUME IN A TABLE</td>
<td>SINGLE-THREADED OR MULTI-THREADED</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

> 500,000 Any

1. Bootstrap Dataverse for the company code.
2. Migrate data to each app outside the initial synchronization.
3. Activate dual-write, and skip the initial synchronization.

**Single-threaded tables**

- Sales tax codes (msdyn_taxcodes)
- Customers V3 (accounts)
- Vendors V2 (msdyn_vendors)
- Warehouses (msdyn_warehouses)
- Product categories (msdyn_productcategories)
- Employment (cdm_employments)
- Position worker assignments (cdm_positionworkerassignmentmaps)
- Warehouse locations (msdyn_inventorylocations)
- Modes of delivery (msdyn_shipvias)
For more consistent availability and performance, limits apply when dual-write is used to write data to Finance and Operations apps and Microsoft Dataverse. These limits control dual-write transactions and are applied at the platform level. They are designed to ensure seamless writes and to help minimize failures.

Finance and Operations apps and Dataverse have many processes that span large numbers of records and complex, multi-table transactions. Each environment has limits on the number of transactions, the number of records per transaction, and transaction time (that is, the time that is required to process the transaction). It’s important that you understand these limits and their effect on the live synchronization capabilities of dual-write.

**Legal Entities**

Live synchronization supports up to 250 legal entities per transaction. This is different from initial synchronization which supports only 40 due to larger data volumes and related operations.

**Transaction patterns**

A process can write data in two different transaction patterns:

- **Single transaction** – All data that is written as part of the process is part of a single transaction.
- **Multiple transactions** – All the data that is written as part of the process is split into multiple transactions.

**Single transaction**

In a single transaction, all data that is written as part of the process is part of a single transaction. If a failure occurs, the whole transaction is rolled back.

A common example is a process that creates multiple invoices in a single transaction. In this case, either all invoices are committed in one transaction or, if an error occurs, all invoices are rolled back. The following code examples create multiple invoices in a single transaction.

```plaintext
// Transaction start
Invoice1.create();
Invoice2.create();
Invoice3.create();
// Additional invoices.
InvoiceN.create();
ttscommit;// Transaction end
```

**Multiple transactions**

In multiple transactions, all the data that is written as part of the process is split into multiple transactions. If a failure occurs, the whole group of multiple transactions isn’t rolled back. Instead, each transaction is independently rolled back or committed. Any transaction that has a failure is rolled back. Any transaction that
has no failure is committed.

A common example is a process that creates multiple invoices. Each invoice is created in a separate transaction, and if an error occurs, only the invoice that is included in a specific transaction is rolled back. The remaining invoices are successfully committed. The following code examples create multiple invoices, one per transaction.

```csharp
  ttsbegin; // Transaction start
  Invoice1.create();
  ttscommit;// Transaction end

  ttsbegin;// Transaction start
  Invoice2.create();
  ttscommit;// Transaction end

  ttsbegin;// Transaction start
  Invoice3.create();
  ttscommit;// Transaction end

  // Additional transactions.

  ttsbegin;// Transaction start
  InvoiceN.create();
  ttscommit;// Transaction end
```

### Transaction time limit

When dual-write is used to write records to Finance and Operations apps or Dataverse, each transaction must be completed within a specific amount of time. If a transaction isn't completed before the transaction time limit is reached, the records aren't committed to Finance and Operations apps and Dataverse by using dual-write. In this case, the records in the transaction are rolled back in both the Finance and Operations environment and the Dataverse environment.

For example, when dual-write is used to sync contract renewals from Finance and Operations apps to Dataverse, the timer begins when the business logic in Finance and Operations apps is completed and the Dataverse process is started. The timer ends when the transaction is committed. The whole time that is spent in Dataverse includes the time that is required to write, and also the time that is required to process the standard and custom plugins. If the transaction exceeds the time limit, the records aren't committed to Dataverse.

The same principle applies when dual-write is used to write data in the other direction, from Dataverse to Finance and Operations apps.

### Dual-write live synchronization limits

The following tables describe the dual-write live synchronization limits that apply when data is written between Finance and Operations apps and Dataverse. These limits are specific to the direction of the data flow: from Finance and Operations apps to Dataverse, or from Dataverse to Finance and Operations apps.

#### From Finance and Operations apps to Dataverse

The following limits apply when data is written from Finance and Operations apps to Dataverse.
### MEASURE | LIMITS
--- | ---
Number of transactions | The total number of transactions that you can perform per day per tenant is governed by service protection API limits that are designed to detect when client applications make extraordinary demands on server resources. For more information, see Service protection API limits.

Number of records per single transaction | 1,000 records
If there are more than 1,000 records in a single transaction, consider splitting that transaction into multiple transactions. For more information, see the Transactions with more than 1,000 records section of this topic.

Transaction time limit | 2 minutes

### From Dataverse to Finance and Operations apps

The following limits apply when data is written from Dataverse to Finance and Operations apps.

### MEASURE | LIMITS
--- | ---
Number of transactions | The number of transactions might be affected by priority-based throttling limits that are designed to help prevent over-utilization of resources and preserve system responsiveness. For more information, see Priority-based throttling.

Number of records per single transaction | A payload size limit on Dataverse limits the number of records that can be transferred. The limit is 116.85 megabytes (MB) per transaction. For more information, see Error: Message size exceeded when sending context to Sandbox. Use of this limit depends on the multiple factors, such entity complexity, the type of columns that is used, and mapped fields. Therefore, the limit can't be expressed as a simple number of records.

If the limit is exceeded, Dataverse rejects the transaction (referred to as a message), and the following error code is used:

*Error Code: -2147220970 Error Message: Message size exceeded when sending context to Sandbox. Message size: ### MB*

If the size of the records in a single transaction exceeds 116.65 MB, consider splitting the transaction into multiple transactions. For more information, see the Transactions with more than 1,000 records section of this topic.

Transaction timeout | 2 minutes

### Transactions with more than 1,000 records

Scenarios where transactions have more than 1,000 records are common. In these scenarios, we recommend that you split single transactions into multiple transactions. The following code examples show how to make multiple transactions, based on record IDs.
ttsbegin; // Transaction start.
   Invoice1.create();
   Invoice2.create();
   // Additional invoices.
   Invoice1000.create();
ttscommit; // Transaction end.

   ttsbegin; // Transaction start.
   Invoice1001.create();
   Invoice1002.create();
   // Additional invoices.
   Invoice2000.create();
ttscommit; // Transaction end.

   ttsbegin; // Transaction start.
   Invoice2001.create();
   Invoice2002.create();
   // Additional invoices.
   Invoice3000.create();
ttscommit; // Transaction end.

   // Additional transactions.
   ttsbegin; // Transaction start.
   Invoice(N)1.create();
   Invoice(N)2.create();
   // Additional invoices.
   Invoice(N+1)000.create();
ttscommit; // Transaction end.

   i = 1;
   committPending = false;
   while /* loop condition */
   {
      if (i==1)
      {
         ttsbegin; // Transaction start.
            committPending = true;
      }
   InvoiceN.create();
   if (i == 1000)
      {
         ttscommit; // Transaction end.
            committPending = false;
            i = 0;
      }
      i++;
   }

   if (committPending == true)
   {
      ttscommit; // Transaction end.
   }
This topic explains how to enable dual-write from Microsoft Dynamics Lifecycle Services (LCS).

Prerequisites

You must complete the Power Platform integration as described in the following topics:

- Power Platform Integration - Enable during environment deployment
- Power Platform integration - Enable after environment deployment

Set up dual-write for new Dataverse environments

Follow these steps to set up dual-write from LCS Environment Details page:

1. On the Environment Details page, expand the Power Platform Integration section.
2. Select the Dual-write application button.
3. Review the terms and conditions, and then select Configure.
4. Select OK to continue.
5. You can monitor the progress by periodically refreshing the environment details page. Setup typically takes 30 minutes or less.
6. When the setup is complete, a message will inform you if the process was successful or if there was a failure. If the setup failed, then a related error message is displayed. You must fix any errors before moving to the next step.
7. Select Link to Power Platform environment to create a link between Dataverse and the current environment's databases. This typically takes less than 5 minutes.
Set up dual-write for an existing Dataverse environment

To set up dual-write for an existing Dataverse environment, you must create a Microsoft support ticket. The ticket must include:

- Your Finance and Operations environment ID.
- Your environment name from Lifecycle Services.
- The Dataverse organization ID or Power Platform Environment ID from Power Platform Admin Center. In your ticket, request that the ID be the instance used for Power Platform integration.

NOTE

You can't unlink environments by using LCS. To unlink an environment, open the Data integration workspace in the Finance and Operations environment, and then select Unlink.

Linking mismatch

It is possible that your LCS environment is linked to one Dataverse instance, while your dual-write environment is linked to another Dataverse instance. This linking mismatch can cause unexpected behavior, and it could end up sending data to the wrong environment. The recommended environment to use for dual-write is the one that is created as part of Power Platform integration, and long term, this will be the only way to establish a link between environments.

If your environment has a linking mismatch, LCS displays a warning on your environment details page similar to "Microsoft has detected that your environment is linked via Dual-write to a different destination than specified in Power Platform Integration, which is not recommended":

If you encounter this error there are two options, based on your needs:
- Unlink and relink dual-write environments (Reset or change linking) as specified on your LCS environment details page. This is the ideal option, because you can run it without Microsoft support.
- If you want to keep your link in dual-write, you can ask for help from Microsoft Support to change the Power Platform integration to use your existing Dataverse environment as documented in the previous section.
Enable dual-write for existing Finance and Operations apps

This set of topics provides step-by-step instructions that explain how to enable dual-write for existing instances of Finance and Operations apps (Microsoft Dynamics 365 Finance and Dynamics 365 Supply Chain Management), and also for a new or existing Dataverse environment.

Step-by-step instructions to enable dual-write for existing instances of Finance and Operations apps and a new or existing Dataverse environment

The process of enabling dual-write has three parts:

1. Make sure that you meet all the system requirements and complete all the prerequisites.
2. Link your Finance and Operations app environment to Dataverse by using the dual-write wizard.
3. Enable the table maps.

Each part is described in a separate topic.

Next steps

System requirements and prerequisites
What regions are available?

Currently, we support dual-write in the following regions:

- Asia
- Australia
- Canada
- Europe
- India
- Japan
- South America
- United Arab Emirates
- United Kingdom
- United States

Verify requirements and grant access

Before you enable dual-write, follow these steps to make sure that you meet the minimum system requirements and to grant access to the apps that must connect to each other. The dual-write health check validates the prerequisites as you complete the dual-write wizard to link a Finance and Operations app environment to a Dataverse environment.

You must set **Enable Dynamics 365 apps** to **Yes** when you set up the environment, as shown in the following image. Alternatively, you can choose a customer engagement app environment that comes with Dataverse and already has **Enable Dynamics 365 apps** set to **Yes**.
1. Validate the platform update and app version.

   Make sure that your Finance and Operations app environment is running Platform update 33 (app version 10.0.9) or later.

   Related health check result:

   App version is up to date
2. Install the dual-write core solution.

The dual-write core solution contains metadata for your table maps and must be installed in your environments.

a. In Power Apps, in the left pane, select **Solutions**.
b. Select **Open AppSource**.
c. Select the **Dual Write Core** solution.
d. Follow the prompts to import the solution.

Related health check result:

*The dual-write core solution was found*

The dual-write core solution contains metadata for your table maps and must be installed in the environment

3. Grant Dataverse access so that it can connect to a Finance and Operations app.

a. Open your instance of the Finance and Operations app, search and navigate to Azure Active Directory applications.
b. Select **New** to add a new client ID row: `6f7d0213-62b1-43a8-b7f4-ff2bb8b7b452`. This row is the application ID for an app that will be used to connect from Dataverse to the Finance and Operations app.
c. Repeat the previous two steps to add another client ID row: `2e49aa60-1bd3-43b6-8ab6-03ada3d9f08b`.

When you've finished, follow these steps to refresh the list of tables:

a. Go to **Workspaces > Data management**, select the **Data entities** tile, and make sure that the entity list is filled in.
b. Go to **Workspaces > Data management**, and select the **Framework parameters** tile. Then, on the **Entity settings** tab (https://<BaseFinanceandOperationsappsURL?>/?cmp=USMF&mi=DM_DataManagementWorkspaceMenu&TableName=DMFDefinitionGroupEntity), select **Refresh entity list**.

Related health check result:
The Dataverse can connect to the Finance and Operations app

Before you can enable dual-write, you must grant access to the apps to connect to each other

App user with id 6f7d0213-62b1-43a8-b7f4-ff2bb8b7b452 exists
App user with id 2e49aa60-1bd3-43b6-8ab6-03ada3d9f08b exists

4. Grant a Finance and Operations app access so that it can connect to Dataverse. Follow the steps in Create an application user, using the following information for applications IDs and security roles.

- **Applications**: Add users to these applications:
  - 00000015-0000-0000-c000-000000000000
  - 2e49aa60-1bd3-43b6-8ab6-03ada3d9f08b

- **Security roles**: Select a preconfigured Security Role to grant a Read privilege with a User scope for each table integrated through dual-write.

  **NOTE**
  Company and currency exchange tables are global in nature and all dual-write users require read access to these 2 tables. All dual-write users will need to be added to the Dual-Write App User security role. In order to allow non-administrator users to create rows in a dual-write enabled table, they will need to be assigned the Dual-Write Runtime User security role.

For instructions on how to create a Security Role, see Create or configure a custom security role.

  **NOTE**
  The root business unit’s default team will become the default owner for all rows integrated through dual-write. Because that team must be assigned a security role, this means that all users in the root business unit will inherit the security role. This means that at the very least, users from that business unit will have read access to all the rows that are owned by that team. If this isn’t the desired behavior, make sure that users are not a member of the root business unit.

**Related health check result:**

The Finance and Operations app can connect to the Dataverse

Before you can enable dual-write, you must grant access to the apps to connect to each other

App user with id 00000015-0000-0000-c000-000000000000 exists
App user with id 2e49aa60-1bd3-43b6-8ab6-03ada3d9f08b exists

5. Provide app consent in the tenant. For dual-write core solution version 1.0.16.0 or above, this step is no longer needed.

**Related health check result:**

Apps in tenant

The required dual-write applications need to be installed in the tenant.

App ID: 6f7d0213-62b1-43a8-b7f4-ff2bb8b7b452
App ID: 2e49aa60-1bd3-43b6-8ab6-03ada3d9f08b

6. Make sure that the dual-write plug-ins are enabled.

This step isn’t usually required, because the plug-ins should be enabled as part of the process of installing the dual-write core solution. However, if the health check fails, follow these steps to manually enable the dual-write plug-ins:

a. Download the Plug-in Registration Tool.

In the Plugin Registration Tool, there should be two plug-in assemblies that are associated with
dual-write: DualWriteRegistration.Plugins and DualWriteRuntime.Plugins. These assemblies have plug-in steps that must be enabled, in order, before dual-write can be used. To view the plug-in steps, expand a plug-in assembly and its plug-in types. All the steps that belong to the dual-write plug-in assemblies should be enabled.

b. To enable a step, select and hold the step (or right-click it), and then select Enable. If no Enable option is available, only a Disable option, the step has already been enabled and doesn't have to be changed.

If the dual-write plug-in assemblies can't be found, import the latest version of the dual-write core solution.

Related health check result:
The dual-write registration and runtime plugins are enabled
To ensure listening into CRUD operations on the Dataverse, the dual-write plugins need to be enabled

7. Install the Dual-write application orchestration solution maps solution.

In Power Apps, in the left pane, select Solutions. Select Open AppSource, and search for the solution that is named Dual-write application orchestration solution. Select the solution, and follow the prompts to import it. After installation, you'll find several new solutions listed under Solutions. For more information, see Solutions overview.

While the dual-write core solution contains metadata for your table maps, the dual-write application orchestration solution covers these additional master data scenarios:

- Customers, products, and vendors.
- End-to-end process flows like prospect to cash.
- On-demand functions like pricing.
- Reference data for ledger, tax, payment terms, and schedules.

Dual-write will continue to expand in the future to support more scenarios including party, project, and hands-on inventory. The framework is extensible and accommodates customer-centric business data exchange through a few additional clicks.
8. Uninstall the Prospect to Cash (P2C) solution.

The P2C solution doesn't work concurrently with dual-write. Therefore, don't install the P2C solution. If it's already installed, you must uninstall it before you enable dual-write.

9. Provide the supported tenant configuration.

Make sure that the Finance and Operations app and Dataverse are installed under the same tenant. Cross-tenant scenarios aren't currently supported.

For dual-write core solution lower than version 1.0.16.0 only

1. In step Step 3b above, create a new client ID row: 33976c19-1db5-4c02-810e-c243db79efde (versus 6f7d0213-62b1-43a8-b7f4-ff2bb8b7b452).

2. Add app consent in the tenant:

   a. Open the following URL, and sign in by using your admin credentials. You should be prompted for consent.

      https://login.microsoftonline.com/common/oauth2/authorize?client_id=33976c19-1db5-4c02-810e-c243db79efde&response_type=code&prompt=admin_consent

   b. Select Accept.

      By selecting Accept, you indicate that you're providing consent to install the app that has application ID 33976c19-1db5-4c02-810e-c243db79efde in your tenant. Dataverse requires this app to communicate with the Finance and Operations app.

Related health check result:

Apps in tenant

The required dual-write applications need to be installed in the tenant.

   App ID: 33976c19-1db5-4c02-810e-c243db79efde
   App ID: 2e49aa60-1bd3-43b6-8ab6-03ada3d9f08b

Next steps

Use the dual-write wizard to link your environments
1. Sign in to the Finance and Operations app environment that you want to link to your Dataverse environment.

2. Go to **Workspaces > Data management**, and select the **Dual Write** tile.

3. Select **New link to environment** to open the **Setup link to Dataverse** wizard.

4. The **Choose environment** page lists all the Dataverse environments where the signed-in user is an environment admin. Select the Dataverse environment to link to, and then select **Next**.
5. Select your legal entities, and then select **Next**.

A health check is run to verify that your system meets the requirements for enabling dual-write. The health check also verifies that all the prerequisites have been completed. If any health check test fails, make sure you've completed all the prerequisites before you move on to the next step.

In the following example, the test about whether access was granted to connect the apps failed. In this case, you must first grant access to connect the apps by creating the appropriate application IDs. You must then rerun the wizard.

6. Review the summary, privacy notice, and consent, and then select **Create**.

You’ve now linked your Finance and Operations app to the Dataverse environment.

**NOTE**

If you don’t see your table maps, or if you see a blank page, be sure to **Apply** the Dual-write application orchestration solution that you installed as part of the system requirements and prerequisites.

7. Apply the dual-write application orchestration solution.
In the Finance and Operations app, on the **Dual-write** page, select **Apply Solution** to apply the table maps that you just downloaded and installed. After you apply the solution, you should see that the default table maps are published.

You've now successfully imported and applied a Microsoft-published dual-write table map solution to your environment.

**Next steps**

**Enable table maps for dual-write**
When you enable a table map for dual-write, it begins at the **Not running** status. The table map then goes through an initialization phase, where it does an initial write by copying pre-existing data on tables on both sides. Finally, when the table is completely enabled, the table map sets the status to **Running**.

During the initialization phase, any pre-existing data that you have is copied as part of the initial write phase.

<table>
<thead>
<tr>
<th>STATUS</th>
<th>DESCRIPTION</th>
<th>AVAILABLE ACTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not running</td>
<td>The table has not yet been enabled for dual-write. Every table begins at the <strong>Not running</strong> status.</td>
<td>Run</td>
</tr>
<tr>
<td>Initializing</td>
<td>The initial write is occurring.</td>
<td>None</td>
</tr>
<tr>
<td>Running</td>
<td>The table has been enabled for dual-write.</td>
<td>Stop, Pause</td>
</tr>
<tr>
<td>Paused</td>
<td>The table is in a paused state, and all new requests are queued.</td>
<td>Run</td>
</tr>
<tr>
<td>Resuming</td>
<td>The table is catching up on rows that were queued while the table was paused.</td>
<td>None</td>
</tr>
</tbody>
</table>

While the status is **Running**, you can pause a table. All changes are then queued until you resume. When you resume, the table goes into "catch-up mode," where all the queued changes are played back.

The following illustration shows an example of a table that is paused.
Entities have several dependent tables. For example, Customer-Contact tables have customer groups and currencies as dependent tables.

Because these are relational apps that have relational data, if you don’t enable the dependent tables, you might encounter errors later. To help prevent these errors, before you enable a table map, you’re provided with a list of the related tables that we recommend that you enable.

Example: Enabling the Customers V3—Contacts table map

When you select a table map (for example, Customers V3—Contacts) and select Run, a dialog box appears before the table map is enabled. This dialog box lists all the dependent tables. You can select the Show related table map(s) option to show all the related table maps. To enable the selected table map and all its related tables, select Run in the dialog box.

NOTE
The behavior is similar when you pause a table. In that case, you have the option to pause all the related tables too.
Criteria for linking tables

To enable table maps for dual-write, you must define an alternative key in Dataverse. The value of the alternative key in Dataverse must match the key that is defined in the Finance and Operations app.

For example, in a Finance and Operations app, CustomerAccount is the key for the Account table.
In Dataverse, `accountnumber` is defined as the key for the Account table.

In the Customers V3 table map, you can see that `accountnumber` is mapped to `CustomerAccount`.
Next steps

Customize table and column mappings
You can increase the number of decimal places that are supported for currency values to a maximum of 10. The default limit is four decimal places. By increasing the number of decimal places, you help prevent data loss when you use dual-write to sync data. The increase in the number of decimal places is an opt-in change. To implement it, you must request assistance from Microsoft.

The process of changing the number of decimal places has two steps:

1. Request migration from Microsoft.
2. Change the number of decimal places in Dataverse.

The Finance and Operations app and Dataverse must support the same number of decimal places in currency values. Otherwise, data loss can occur when this information is synced between apps. The migration process reconfigures the way that currency and exchange rate values are stored, but it doesn't change any data. After the migration is completed, the number of decimal places for currency codes and pricing can be increased, and the data that users enter and view can have more decimal precision.

Migration is optional. If you might benefit from support for more decimal places, we recommend that you consider migration. Organizations that don't require values that have more than four decimal places don't have to migrate.

**Requesting migration from Microsoft**

Storage for existing currency columns in Dataverse can't support more than four decimal places. Therefore, during the migration process, currency values are copied to new internal columns in the database. This process occurs continuously until all data has been migrated. Internally, at the end of migration, the new storage types replace the old storage types, but the data values are unchanged. The currency columns can then support up to 10 decimal places. During the migration process, Dataverse can continue to be used without interruption.

At the same time, exchange rates are modified so that they support up to 12 decimal places instead of the current limit of 10. This change is required so that the number of decimal places is the same in both the Finance and Operations app and Dataverse.

Migration doesn't change any data. After the currency and exchange rate columns are converted, admins can configure the system to use up to 10 decimal places for currency columns by specifying the number of decimal places for each transaction currency and for pricing.

**Request a migration**

To make this feature available, email **CDSExpandDecimal@microsoft.com**, and include the following information:

- **Subject**: Request to enable expanded decimal support for <organizationID>
- **Body**: I would like to enable expanded decimal support for my org <organizationID>.

A Microsoft representative will contact you within two to three business days for the next steps.

When you request a migration, you should be aware of the following details and plan for them accordingly:

- The time that is required to migrate the data depends the amount of data in the system. Migration of large databases can take several days.
- The size of the database temporarily increases while the migration is running, because additional space is
needed for indexes. Most of the additional space is freed when the migration is completed.

- During the migration process, if errors occur that prevent the migration from being completed, the system raise alerts to Microsoft Support, so that Support staff can intervene. However, even if errors occur during the migration, Dataverse remains fully available for regular use.
- The migration process isn’t reversible.

**Changing the number of decimal places**

After the migration is completed, Dataverse can store numbers that have more decimal places. Admins can choose how many decimal places are used for specific currency codes and for pricing. Users of Microsoft Power Apps, Power BI, and Power Automate can then view and use numbers that have more decimal places.

To make this change, you must update the following settings in Power Apps:

- **System Settings: Currency precision for pricing** – The Set the currency precision that is used for pricing throughout the system column defines how the currency will behave for the organization when Pricing Precision is selected.
- **Business Management: Currencies** – The Currency Precision column lets you specify a custom number of decimal places for a specific currency. There is a fallback to the organization-wide setting.

There are some limitations:

- You can’t configure the currency column on a table.
- You can specify more than four decimal places only at the Pricing and Transaction Currency levels.

**System Settings: Currency precision for pricing**

After migration is completed, admins can set the currency precision. Go to **Settings > Administration**, and select **System Settings**. Then, on the **General** tab, change the value of the **Set the currency precision that is used for pricing throughout the system** column, as shown in the following illustration.

![System Settings](image)

**Business Management: Currencies**

If you require that the currency precision for a specific currency differ from the currency precision that is used for pricing, you can change it. Go to **Settings > Business Management**, select **Currencies**, and select the currency to change. Then set the **Currency Precision** column to the number of decimal places that you want, as shown in the following illustration.
The number of decimal places that can be configured for specific currency columns is limited to four.
The columns that indicate sales order status have different enumeration values in Microsoft Dynamics 365 Supply Chain Management and Dynamics 365 Sales. Additional setup is required to map these columns in dual-write.

**columns in Supply Chain Management**

In Supply Chain Management, two columns reflect the status of the sales order. The columns that you must map are **Status** and **Document Status**.

The **Status** enumeration specifies the overall status of the order. This status is shown on the order header.

The **Status** enumeration has the following values:

- Open Order
- Delivered
- Invoiced
- Cancelled

The **Document Status** enumeration specifies the most recent document that was generated for the order. For example, if the order is confirmed, this document is a sales order confirmation. If a sales order is partially invoiced, and then the remaining line is confirmed, the document status remains **Invoice**, because the invoice is generated later in the process.

The **Document Status** enumeration has the following values:

- Confirmation
- Picking List
- Packing Slip
- Invoice

**columns in Sales**

In Sales, two columns indicate the status of the order. The columns that you must map are **Status** and **Processing Status**.

The **Status** enumeration specifies the overall status of the order. It has the following values:

- Active
- Submitted
- Fulfilled
- Invoiced
- Cancelled

The **Processing Status** enumeration was introduced so that the status can be mapped more accurately with Supply Chain Management.

The following table shows the mapping of **Processing Status** in Supply Chain Management.
<table>
<thead>
<tr>
<th>PROCESSING STATUS</th>
<th>STATUS IN SUPPLY CHAIN MANAGEMENT</th>
<th>DOCUMENT STATUS IN SUPPLY CHAIN MANAGEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Open Order</td>
<td>None</td>
</tr>
<tr>
<td>Confirmed</td>
<td>Open Order</td>
<td>Confirmation</td>
</tr>
<tr>
<td>Picked</td>
<td>Open Order</td>
<td>Picking List</td>
</tr>
<tr>
<td>Partially Delivered</td>
<td>Open Order</td>
<td>Packing Slip</td>
</tr>
<tr>
<td>Delivered</td>
<td>Delivered</td>
<td>Packing Slip</td>
</tr>
<tr>
<td>Partially Invoiced</td>
<td>Delivered</td>
<td>Invoice</td>
</tr>
<tr>
<td>Invoiced</td>
<td>Invoiced</td>
<td>Invoice</td>
</tr>
<tr>
<td>Cancelled</td>
<td>Cancelled</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

The following table shows the mapping of Processing Status between Sales and Supply Chain Management.

<table>
<thead>
<tr>
<th>PROCESSING STATUS</th>
<th>STATUS IN SALES</th>
<th>STATUS IN SUPPLY CHAIN MANAGEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Active</td>
<td>Open Order</td>
</tr>
<tr>
<td>Confirmed</td>
<td>Submitted</td>
<td>Open Order</td>
</tr>
<tr>
<td>Picked</td>
<td>Submitted</td>
<td>Open Order</td>
</tr>
<tr>
<td>Partially Delivered</td>
<td>Active</td>
<td>Open Order</td>
</tr>
<tr>
<td>Partially Invoiced</td>
<td>Active</td>
<td>Open Order</td>
</tr>
<tr>
<td>Partially Invoiced</td>
<td>Fulfilled</td>
<td>Delivered</td>
</tr>
<tr>
<td>Invoiced</td>
<td>Invoiced</td>
<td>Invoiced</td>
</tr>
<tr>
<td>Cancelled</td>
<td>Cancelled</td>
<td>Cancelled</td>
</tr>
</tbody>
</table>

**Setup**

To set up the mapping for the sales order status columns, you must enable the IsSOPIntegrationEnabled and isIntegrationUser attributes.

To enable the IsSOPIntegrationEnabled attribute, follow these steps.

1. In a browser, go to `https://<test-name>.crm.dynamics.com/api/data/v9.0/organizations`. Replace `<test-name>` with your company’s link to Sales.

2. On the page that is opened, find `organizationid`, and make a note of the value.
3. In Sales, open the browser console, and run following script. Use the `organizationid` value from step 2.

```javascript
Xrm.WebApi.updateRecord("organization",
"d9a7c5f7-acbf-4aa9-86e8-a891c43f748c", {"issopintegrationenabled": true}).then(
    function success(result) {
        console.log("Account updated");
        // perform operations on row update
    },
    function (error) {
        console.log(error.message);
        // handle error conditions
    }
);
```

4. Verify that `IsSOPIntegrationEnabled` is set to `true`. Use the URL from step 1 to check the value.

To enable the `isIntegrationUser` attribute, follow these steps.

1. In Sales, go to `Setting > Customization > Customize the System`, select `User table`, and then open `Form > User`. 
2. In Field Explorer, find **Integration user mode**, and double-click it to add it to the form. Save your change.

3. In Sales, go to **Setting > Security > Users**, and change the view from **Enabled Users** to **Application Users**.
4. Select the two entries for **DualWrite IntegrationUser**.

5. Change the value of the **Integration user mode** column to **Yes**.
Your sales orders are now mapped.
You can filter intercompany orders so that the **Orders** and **OrderLines** tables aren’t synced. In some scenarios, the intercompany order details aren’t required in a customer engagement app.

Each standard Dataverse table is extended through references to the **IntercompanyOrder** column, and the dual-write maps are modified so that they refer to the additional columns in the filters. Therefore, the intercompany orders are no longer synced. This process helps prevent unnecessary data in the customer engagement app.

1. **Extend the CDS Sales Order Headers** table by adding a reference to the **IntercompanyOrder** column. This column is filled in only on intercompany orders. The **IntercompanyOrder** column is available in the **SalesTable** table.

2. **After CDS Sales Order Headers** is extended, the **IntercompanyOrder** column is available in the mapping. Apply a filter that has **INTERCOMPANYORDER = ""** as the query string.

3. **Extend the CDS Sales Order Lines** table by adding a reference to the **InterCompanyInventTransId** column. This column is filled in only on intercompany orders. The **InterCompanyInventTransId** column is available in the **SalesLine** table.

4. **After CDS Sales Order Lines** is extended, the **IntercompanyInventTransId** column is available in the
5. Repeat steps 1 and 2 to extend the Sales Invoice Header V2 table and add a filter query. In this case, use \( \text{INTERCOMPANYORDER} == "\" && \text{SALESORDERNUMBER} != "\" \) as the query string for the filter.

6. Repeat steps 3 and 4 to extend the Sales Invoice Lines V2 table and add a filter query. In this case, use \( \text{INTERCOMPANYINVENTTRANSID} == "\" \) as the query string for the filter.

7. The Quotations table doesn't have an intercompany relationship. If someone creates a quotation for one of your intercompany customers, you can use the CustGroup column to put all those customers into
one customer group. You can extend the header and lines by adding the **CustGroup** column, and then filter so that the group isn’t included.

8. After **Quotations** is extended, apply a filter that has \[\text{CUSTGROUP} \neq \text{"company"}\] as the query string.
The out-of-box table maps have predefined table and column mappings that enable the flow of data between two apps. In this way, they serve as "blueprints." However, because every business is different, the default table maps might sometimes not be enough. Therefore, dual-write fully supports customization by providing ways to change table maps and column mappings.

**Customize column mappings, add transforms, and enable filtering**

1. In your Finance and Operations app, on the Dual-write page, on the Table mappings tab, select the table map to customize.

   **NOTE**
   Before you change table mappings, they must be stopped (not running). Otherwise, your changes won't be saved.

2. On the Table mappings tab, you can customize a column by selecting a new or custom column from either the Finance and Operations app or Dataverse.

   ![Customize Column Mappings](image.png)

3. You can customize the synchronization direction (unidirectional or bidirectional) and add transforms by selecting the map type.
The following table describes the available synchronization directions.

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>➡️⬅️</td>
<td>Bidirectional column assignment</td>
</tr>
<tr>
<td>➡️</td>
<td>Bidirectional column assignment that uses transforms</td>
</tr>
<tr>
<td>➢</td>
<td>Unidirectional column assignment (left to right)</td>
</tr>
<tr>
<td>➢</td>
<td>Unidirectional column assignment (right to left)</td>
</tr>
<tr>
<td>➡️</td>
<td>Unidirectional column assignment that uses transforms (left to right)</td>
</tr>
<tr>
<td>⬅️</td>
<td>Unidirectional column assignment that uses transforms (right to left)</td>
</tr>
</tbody>
</table>

The following table describes the available transform types.

<table>
<thead>
<tr>
<th>TRANSFORM TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Default values are values that are applied to destination columns when no source column value is available. Use default values for columns that are required on the destination table when you have no corresponding source column.</td>
</tr>
<tr>
<td>Value map</td>
<td>Value maps define how values that are present in one table should be mapped to values in the other table.</td>
</tr>
</tbody>
</table>

4. You can add a new column by selecting **Add mapping** and then selecting an existing or custom column in the list.

The following illustration shows an example where a new **birthdate** column is being added.
5. When you’ve finished customizing the column mappings, select **Save**. Then follow the prompts to specify a publisher and a version number.

**Filter your data**

Dual-write lets you filter data by using Open Data Protocol (OData) filter expressions for Dataverse. For the Finance and Operations app, filtering resembles range expressions that are used in the query range.

1. On the table mapping page, select the filter button (funnel symbol).
2. In the **Edit query** dialog box, specify your filters. In this example, the filter that is specified will return only accounts where the account type equals 3.

The following table shows some examples of filter expressions.

<table>
<thead>
<tr>
<th>FILTER</th>
<th>DATVERSE</th>
<th>FINANCE AND OPERATIONS APPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>String field like</td>
<td><code>startswtih(name, 'A')</code></td>
<td><code>(name like 'A*')</code></td>
</tr>
<tr>
<td>String field not like</td>
<td><code>not contains(name, 'A')</code></td>
<td><code>(!((name like 'A')))</code></td>
</tr>
<tr>
<td>Enumeration fields</td>
<td><code>AccountType == '3'</code></td>
<td><code>(AccountType == AccountType::Customer)</code></td>
</tr>
<tr>
<td>Dates</td>
<td><code>TransactionDate le '2021-06-23'</code></td>
<td><code>((TransactionDate &lt;= '2021-06-23'))</code></td>
</tr>
<tr>
<td>Multiple criteria combined</td>
<td><code>numberofemployees gt 1000 and numberofemployees le 2000</code></td>
<td><code>((numberofemployees &gt; 1000) &amp;&amp; (numberofemployees &lt;= 2000))</code></td>
</tr>
</tbody>
</table>

For more examples that show how to use expressions in query ranges, see [Using Expressions in Query Ranges](#).

Currently, we do not support nested lookups in dual-write source filter. Only standard filter operators...
directly against table columns are supported. For more examples, see Standard filter operators.

Add new table maps

Although Microsoft is continuing to add new tables, you can also add standard or custom table maps.

The following example shows how to add a new table map that is named **Address books**.

1. In the Finance and Operations app, on the **Dual-write** page, select **Add table map**.

![Add table map screenshot](image)

**NOTE**

When you create a new solution that uses these modified table maps, you must specify the same publisher:

2. Confirm the table maps that you just modified and added. Be sure to enable and test them, to ensure that they work as you expect.
Next steps

Error management and alert notifications
Dual-write provides out-of-box maps for some business processes. However, there might be scenarios where you need additional fields, maps, or transformations. The dual-write platform is extensible. You can create custom maps and extend existing maps with custom fields to sync data between Finance and Operations apps and Microsoft Dataverse. This topic provides guidance and best practices for these customizations.

Before you customize any maps, you should be familiar with the tasks in Customize table and column mappings.

**Guidance when the entity is in both the Finance and Operations app and Dataverse**

If the entity is in both environments, create a dual-write map.

- Integration keys for the entity should match in both environments. If the entity key isn’t available on either side, be sure to create entity keys. The integration key fields should be mapped to each other in the map.
- The company field should not be present in the mapping if the entity is legal entity-specific, because the company field will already be part of the key. For an example, review the Customer groups (msdyn_customergroups) entity mapping.
- Add filters to either the Finance and Operations environment or the Dataverse environment to trigger the dual-write map only on specific criteria. The Customers V3 - Accounts or CDS Contacts V2 (Contacts) map has several filters that you can use as examples.

**Guidance when the entity is in the Finance and Operations app only**

If the entity is in the Finance and Operations app only, create a new entity in Dataverse, and then add a map.

- If the Finance and Operations entity contains data that is legal entity-specific, be sure to add a lookup field to cdm_companies in the new Dataverse entity. If the Finance and Operations entity is global, a field for the company isn't required in the Dataverse entity.
- Add keys to the Dataverse entity to mimic the Finance and Operations entity key. Dual-write requires the same entity keys in both the Finance and Operations environment and the Dataverse environment. The key fields on the Finance and Operations app and Dataverse should be mapped to each other. Don’t add the company field in the mapping. For an example, review the Vendors V2 - msdyn_vendors mapping.

**Guidance when the entity is in Dataverse only**

If the entity is in Dataverse only, create a new entity in the Finance and Operations environment, and then add a map.

- Create the new entity, and include all the required fields. Make sure that the entity is enabled for Data management and public, so that it can be consumed by OData (Open Data Protocol). For more information about how to create a new entity, see Build and consume data entities.
- If data in the Finance and Operations app should be legal entity-specific, be sure to add a lookup field to cdm_companies in the Dataverse entity. If the Finance and Operations entity is global, a field for the company isn't required in the Dataverse entity.
- Make sure that both entities have the entity key fields. Dual-write requires the same entity keys in both the Finance and Operations environment and the Dataverse environment. The key fields on the Finance and Operations app and Dataverse should be mapped to each other. Don’t add the company field in the
Add attributes to a mapping

If the entities exist in both environments and are mapped, you can add attributes to the map. For more information, see Customize table and column mappings.

Create and update operations don't trigger the synchronization of attributes to Dataverse

In some situations, the entities exist in both environments, but create and update operations don't trigger the synchronization of attributes to Dataverse. Go to the BusinessEventsDefinition table either by using SQL on the Finance and Operations virtual machine (VM) or by using the table browser. Make sure that there is a record for the combination of the affected table that has an updated date (in the RefTableName field) and the entity name (in the RefEntityName field). An example is shown in the following image.

Guidance when entities aren't available in either the Finance and Operations app or Dataverse

If the entities don't exist in either environment, you can create tables in both environments and then create the app by following these steps.

1. In Dataverse, create a new table that has all the required fields. Follow the steps in Create a custom table. If the table should store legal entity–specific data, be sure to add a lookup field to cdm_companies in the new Dataverse table. If the table stores global data, a field for the company isn't required in the Dataverse table.

2. In the Finance and Operations app, create a new entity that has all the required fields. Make sure that the entity is enabled for Data management and public, so that it can be consumed by OData. For more information about how to create a new entity, see Build and consume data entities.

3. To enable table maps for dual-write, define an alternative key in the Dataverse table. The value of the alternative key in Dataverse must match the key that is defined in the Finance and Operations app. For example, in the Finance and Operations app, CustomerAccount is the key for the Account table, as shown in the following illustration.
In Dataverse, `accountnumber` is defined as the key for the `Account` table, as shown in the following illustration.

If you review the `Customers V3` table map, you can see that `accountnumber` is mapped to `CustomerAccount`.
Best practices for dual-write

- Changes must be in a transaction. It's important that you evaluate the number of records per transaction, based on your table design. It's also important that you evaluate how transaction blocks are structured as part of the process in X++. Two examples are shown here.

```plaintext
ttsbegin;
// Transaction start
table1record1.insert();
table1record2.insert();
table1record3.insert();
table1recordN.insert();
ttscommit;
// Transaction end
```

```plaintext
ttsbegin;
// Transaction start
while ( // loop conditions// )
{
  table1recordN.insert();
}
ttscommit;
// Transaction end
```

- The following items aren't handled by business events. Therefore, they aren't handled by dual-write.
  - The `doUpdate` method
  - The `doInsert` method
  - Set-based operations (`insert` and `update`)
  - Records where `skipBusinessEvents(true)` is marked
- Business events must be registered for the data source that is mapped. Data sources aren't tracked if they are outer-joined and marked as read-only in the Finance and Operations app.
- Changes are triggered only if the modifications are on the mapped fields in the Finance and Operations app. In customer engagement apps, all field modifications trigger dual-write synchronization.
Every filter evaluation should provide a valid result.

Data sources aren’t tracked if they don’t have any fields that are mapped.

Entity relationships in the Finance and Operations app must indicate to dual-write that the two entities are linked, and that relationships exist between the two records in the same transaction. Dual-write batching depends on entity relationships that are explicitly defined and considered to sequence the record insertion if both the parent record and child record are part of the same transaction on related entities. If a business process in the Finance and Operations app involves several entities and must be enabled as batch mode in the customer engagement app, dual-write expects the relationships to be identified and defined on the entity. The following illustration shows the relationship between Sales Order header V2 and Sales Order Line V2.

To help prevent performance issues, avoid using a large number of data sources in dual-write data tables that raise multiple events for a record change. Don’t map unwanted fields in dual-write, and avoid excessive business logic on tables and entities.

If a custom entity in the Finance and Operations app is company-specific (that is, the primary company context property of the entity is set to DataAreaId, as shown in the following illustration), the related Dataverse table should have a company lookup as one of the key columns. Mapping between the shared entity and the company-specific entity isn’t allowed. You can determine whether a Finance and Operations app entity is shared or company-specific by looking at the entity property in Visual Studio Application Explorer. For more information, see Cross company behavior of Data entities.

Filter guidance for maps

You can apply filters to both Finance and Operations entities and Dataverse tables. Filters should be applied only on fields that are present on the dual-write maps. Verify the filter results before you add them to dual-write maps.

For Finance and Operations entities, you can verify filter expressions by using the following code example in an X++ runnable class. Replace the expression and the entity name, and run the class.
```javascript
var entityName = "PROJECTENTITY";
var filterExpression = '(ParentProject == "")';
Query query = new Query();
query.literals(NoYes::Yes);
QueryBuildDataSource qbd = query.addDataSource(tablename2id(entityName));
qbd.addRange(fieldname2id(qbd.table(),identifierStr(RecVersion))).
  value(filterExpression);
qbd.addSelectionField(fieldname2id(qbd.table(),identifierStr(RecId)));
QueryRun qRun = new QueryRun(query);
// This provides the actual SQL statement to execute
var actualSqlStatement = query.getSQLStatement();
while(qRun.next())
{
  var rec = qRun.get(tableName2Id(entityName));
}

For Dataverse tables, you can verify filter expressions by adding the expression as a filter condition on the OData expression.

https://<Env URL>/api/data/v9.0/<TableName>?$filter=<fieldname> eq <value>

For more information about filters, and more examples, see Examples and patterns for filtering.
```
This article describes how to select multiple table maps, view a list of dependent table maps, enable the table maps and all of its related tables, and copy pre-existing data.

As part of day to day operations, there may be a need to bulk handle table maps. For example, you may want to simultaneously enable or pause a set of table maps. Instead of doing this one by one, which is cumbersome and time consuming, you can now enable, pause, resume, or stop more than one table map at the same time in the dual-write list page.

As part of enabling multiple table maps, you also get to view the list of all the dependent table maps by selecting **Show related table map(s)**.

To enable the selected table map and all its related tables, select **Run**. If you want to copy the pre-existing data
for the selected table maps or its dependents, select the corresponding Initial sync check box. Alternatively, remove one or more of the related tables by clearing the corresponding check box. You can also drag and drop the table maps to change the order in which the maps will be synced.
The dual-write wizard enables you to add or remove a company or legal entity after dual-write has been set up. You can do this without having to unlink and relink your dual-write environment.

The wizard enables you to link your Finance and Operations apps to Dataverse environments. As part of this wizard, you also can select one or more companies or legal entities. The company or legal entity list doesn’t remain static and is constantly changing. This is because you may need to add new companies, especially as part of a phased rollout or acquisitions. Until now, you were unable to add a company or legal entity without system down-time, which required you to unlink and relink your environment. All of this can be expensive, especially because of pre-existing data. With this feature, you can add a company in a live environment without the need to unlink your existing dual-write environment.

Add a company or legal entity after dual-write has been set up

Follow these steps to add a company or legal entity after dual-write has been set up.

1. On the Dual-write table map list page, select the Environment details button.

2. On the Legal entities tab, you see the company that you selected as part of the dual-write wizard to link environments. In this example, the company is USMF.
3. Select **Add legal entity** to add one or more companies to dual-write. In this example, GBSI. Select **Save**.

At this point, the legal entities start updating. The table maps that are currently running or paused go through the initial write process by copying pre-existing data. Until the process is completed, we recommend that you do not perform any actions to modify your table maps.
NOTE
This operation may fail if either of the following conditions are true:

- You add or remove a new company when one or more table maps is already in the Initial writes state. This the process where the system is copying pre-existing data.
- You remove a company when one or more table maps is in the Paused state.

4. After the process is complete, a banner displays informing you that the legal entities have been updated successfully. You can now resume updates to your table maps.
You can pause table maps, either manually or automatically via rules. By pausing table maps, you help ensure business continuity, especially during planned or unplanned maintenance. While the app is being maintained, users can continue to do their work and create records.

When you pause a table map that is in the **Running** state, all records that have been created or updated are queued until you resume the table map. The queued records are stored in secure Microsoft Azure storage. They are then played back when you resume the table map and put it back into the **Running** state.

**NOTE**

While a table map is in the **Paused** state, there are limits on the number of records that you can queue and the amount of time that you can queue them for. Whichever limit occurs first will apply. The process starts with soft limits and eventually enforces harder limits to help protect you from exceeding the storage limits.

Records that have been created or updated for a table map that is in the **Paused** state can be viewed on the **Queued records** tab for each table map.

The **Total queued record count** line shows the total number of records that have been queued for a given table map. You can select **Load more** to view additional records in the paginated view. You can also filter the records by the integration key.

When you resume the table map, its state is changed from **Paused** to **Running**, and the records are written from the queue to the destination app. Some records might error out and fail to be written for various reasons, such as business validations in the destination app. In these cases, the records will remain in the queue and can be viewed on the **Catch-up errors** tab. For more information, see Catch-up errors from pausing a table map.
Microsoft has invested lots of time and effort into making dual-write resilient to errors. However, if you encounter an issue while or after you enable table maps for dual-write, you can select specific table maps to get a consolidated view of all the activities and errors for them. This consolidated view includes error logs. The goal is to help you during troubleshooting by providing a single view of the activities for a table map.

Consolidated error management

The activity log provides a chronological list of events that a specific table map goes through from the Not Running status to the Running status. For example, the list can include mappings that are created, updates of column mappings, and mappings that are run. Additionally, if errors occur, you can download the logs to get the next level of details.

Re-running execution for Initial sync

If you encounter issues while you copy pre-existing data between Finance and Operations apps and Dataverse, the Initial sync details tab provides a count of the errors.

Clicking on the individual project will show you the direction in which the sync failed (Finance and Operations app to Dataverse or vice-versa) and details of why it failed. You can choose to fix the underlying issues and then select Re-run execution which retries the entire execution, along with the records that failed or errored out in the last sync. Once this completes, initial sync is completed and the table returns to the Running state. There may be cases where you want to ignore the errors and add new incremental data. In these cases, you can select Rerun execution without errors, which lets you add new data and not retry the errored records.
Catch-up errors from pausing a table map

When you resume a table map after pausing, some records might error out and fail to write due to various reasons including business validations on destination app. In these cases, the records will continue to remain in the queue and can be viewed under the Catch-up errors tab.

The detailed Error message will help you fix the underlying issue after which you could Retry selected records or Retry All records. Once the retry is successful, Retry status will be marked as Completed.
Alert notifications

As an admin, you can create one or more alert settings to handle cases of planned or unplanned maintenance. For example, you can set up the dual-write system to notify you by email if a specific error threshold is reached because of, for example, network errors. The dual-write system can also take action on your behalf. For example, it can pause or stop dual-write.

The following illustration shows an example where dual-write will be paused if 10 errors of the Application error type occur within 15 minutes.

By selecting Create alert settings, you can create more alerts. You can also select whether notifications should be sent to an individual or a group, and whether the dual-write system should take any action on your behalf. To send alerts to a group, enter the values separated by commas, for example, "id1@contoso.com, id2@contoso.com".

---

NOTE

Errored records will be available in the queue for 7 days after which will the queue will be purged. In some cases, you may no longer need these records and they can be deleted from the queue.
In order for your alerts to take effect, you need to restart your table maps

This feature is especially useful if there is unplanned maintenance. For example, one of the apps becomes unavailable and, based on your defined thresholds, dual-write goes into a paused state where all new requests are queued (that is, they aren’t lost). After you fix the underlying issue, and both apps are running smoothly, you can resume from the paused state. The updates will then be read back from the queue and written to the recovered app.

**Next steps**

Application lifecycle management
By making dual-write solution-aware, you enable basic application lifecycle management (ALM) capabilities, such as transportation and backup/restore of dual-write table maps across environments. You also enable scenarios where you can get solutions that are published by Microsoft or an independent software vendor (ISV) from AppSource.

What is a dual-write solution?

A dual-write solution can contain one or more dual-write table maps. These maps can be imported into your environment (by selecting Solutions in Microsoft Power Apps). They can also be exported to other environments as a package. You can import Microsoft-published or ISV-published table maps from AppSource, modify them in your test environment, test them, and then, when they are ready, export them to your production environment. Additionally, you can publish your solution through AppSource, so that other people can use it.

There are two types of solutions: managed and unmanaged.

A managed solution can’t be modified, and it can be uninstalled after it’s imported. When you import an unmanaged solution, you add all the components of that solution into your environment. When you import an unmanaged solution that contains components that you’ve already customized, your customizations are overwritten by the customizations in the imported unmanaged solution.

For more information about solutions, see the solutions overview.

Install the dual-write core solution

The dual-write core solution contains metadata for your table maps and must be installed in your environments.

1. In Power Apps, in the left pane, select Solutions.
2. Select Open AppSource, and search for the solution that is named Dual Write Core.
3. Follow the prompts to import the solution.
Install the dual-write table maps solution

1. In Power Apps, in the left pane, select **Solutions**.

2. Select **Open AppSource**, and search for the solution that is named **Dataverse Add-in for Finance and Operations package**.

3. Follow the prompts to import the solution.

4. In the Finance and Operations app, on the **Dual-write** page, select **Apply Solution** to apply the table maps that you downloaded and installed. After you apply the solution, you will see that the default table maps are published.

![Apply Solution](image)

You've now successfully imported and applied a Microsoft-published dual-write table maps solution to your environment.

Import table maps through a dual-write solution and apply them to your environment (New environments)

This section explains how to import table maps from AppSource and apply them to your environment.

1. Import the dual-write core solution.
   
a. Create a new dual-write environment (a Finance and Operations app environment and a Dataverse environment).
   
b. Follow the instructions in the **Install the dual-write core solution** section earlier in this topic to install the dual-write core solution from AppSource in Power Apps.
   
c. Verify that the dual-write core solution is listed under **Solutions** in Power Apps.

2. Import the Microsoft-published or ISV-published table maps solution.
a. Follow the instructions in the Install the dual-write table maps solution section to download and install the Microsoft-published or ISV-published table maps from AppSource in Power Apps.

b. Verify that the table maps solution is listed under Solutions in Power Apps.

3. Apply the dual-write table maps solution to your Finance and Operations app environment.

   Apply the solution that you downloaded by selecting Apply Solutions on the Dual-write page in the Finance and Operations app, as described in the Install the dual-write table maps solution section.

Update table maps and export them to other environments as a solution

This section explains how to export your customized table maps as a solution, use it as a backup, and move the artifacts across environments and/or publish them to AppSource.

Customize your table maps

The first step is to customize your table maps by modifying existing table maps and adding a new table map.

1. In the Finance and Operations app, on the Table mappings tab, customize the mappings for the default table map that you just installed by using a solution. To add a new table map, select Add Table. In both cases, when you save the table map, you're prompted to specify the publisher and the version number.

   The following figure shows how to add a new column that is named birthday to the contacts - CDS Contacts V2 table map and select the default publisher.
NOTE
When you create a new solution by using these modified table maps, you must specify the same publisher.

The following figure shows how to add a new table map that is named Address books.

2. Confirm the table maps that you just modified and added. Be sure to enable and test them, to ensure that they work as you expect.
Create a new dual-write solution and add your components (Customized table maps)

Now that you’ve customized your mappings and added new mappings, the next step is to create a new dual-write solution and add the table maps to it.

1. In Power Apps, in the left pane, select **Solutions**, and then select **New solution** to create a solution. For this example, the solution is named **MyCustomTableMaps**. Be sure to select the same publisher that you selected in previous steps.

2. Select **Create**. The new solution appears on the **Solutions** list page.
3. Now that you've created your dual-write solution, you can add the customized table maps that you created in previous steps. Select the MyCustomTableMaps solution that you just created, select Add existing, point to Other, and then select Dual Write table map.

4. In the list, select the customized table maps, and add them to the solution. The solution should now contain your customized tables.

You've now customized your tables and put them into a solution.

Export and publish your solution
After you run the solution checker and make sure that there are no issues, you export the solution that you created and publish the changes.

1. In the list of solutions, select your solution, and then select Export.
2. Update the version number, and select whether you want to export the solution as a managed or unmanaged solution. (We recommend that you export it as a managed solution.) Then select Export.

3. Before you export, select Publish all changes, and then select Check for issues. When you’ve finished, select Next to publish all your changes.

The solution, together with all its components, is exported to a zip file.
You've now customized your tables, added them to a new solution, and created a solution file that can be imported and applied to other environments. (This capability can be useful if you want to move table maps between test and production environments.) In a similar way, you can create a backup of all your table maps by adding them to a solution and exporting the solution as a package. That package can then be imported into any environment to restore the table maps.

For information about how to publish the package to AppSource, see Publish your app on AppSource.

**Test your exported solution package**

You can test your exported solution package by importing and applying it to another environment.

1. In Power Apps, select **Import** to import the package into a new environment.

2. Apply the solution that you just imported to the environment.
3. Verify that the two customized table maps appear on the dual-write table maps list page.

4. Make sure that the customizations from previous steps are preserved.
Use the table map version

Sometimes, a solution might contain different implementations of a table map. For example, the version of the contacts - CDS Contacts V2 table map might have a different publisher or a newer version number. In these cases, you can use the Table Map version button to select which table map you want to use in your environment.

Upgrade existing dual-write environments for solution awareness (Existing environments)

1. Import the dual-write core solution.
   a. Follow the instructions in the Install the dual-write core solution section earlier in this topic to import the dual-write core solution from AppSource into Power Apps.
   b. Verify that the dual-write core solution is listed under Solutions in Power Apps.
2. Upgrade the table maps.
   You’ll see a notification prompting you to upgrade.
Select Upgrade table maps at the top of the page.

The upgrade takes a few minutes. When it's completed, you receive a notification.
In Finance and Operations apps, global tables are not associated with a company or legal entity. For these tables, you can specify a team and not use a default team as owner when writing to Microsoft Dataverse using dual-write.

By default, when you enable dual-write, the root business unit’s default team will become the default owner for all rows integrated through dual-write. This may not be what you want when you want to limit access to these records to just a subset of users. It’s not uncommon for an organization to have multiple departments defined by business units with corresponding teams under them. You don’t want all users of the default team to have access to all the records integrated via dual-write. In these situations, you can specify a different team for each global table as an owner for these records.

**Change the owning team**

When you select a global table map such as Global Products, under Table mappings, you can view the list of teams in the Update owning team section. By default, dual-write uses the default team, which is indicated by a blank value. After you create a new map with a new version, you change the default behavior by picking a new team from the owning team list and then save the new value. The **Owning team** field is shown in the following screenshot.

![Screenshot showing the owning team field](image)

After you set the owning team, it works across both initial and live sync. The owning team setting does not transfer across environments.

**NOTE**

After you set the owning team, it works across both initial and live sync. The owning team setting does not transfer across environments.
View the owning team after initial sync

After you run the initial sync, the owner is shown in the integrated records. In the following screenshot, the owner has changed from **dwteam** to **Sales** for the integrated records in **Global Product** table map.
When you unlink and relink dual-write connection between environments, you need to delete the data from the key tables. This requirement applies to sandbox, production, and user acceptance test (UAT) environments during activities like backup and restore. This topic describes how to unlink, delete the data in the key tables, and then relink the dual-write environments.

The mappings are preserved when you unlink and relink, because the mappings are stored in Dataverse.

**Scenario: Dual-write is enabled between production environments**

In this scenario, dual-write is enabled between Finance and Operations and Dataverse production environments. You want to back up the Finance and Operations production environment (source) and restore it to Finance and Operations UAT environment (destination). Once you restore, follow these steps on the Finance and Operations UAT environment:

1. Stop all table maps.
2. Unlink the dual-write connection as the Finance and Operations UAT environment will be pointing towards Dataverse production environment.
3. Delete the data from the key tables:
   - **DualWriteProjectConfiguration**
   - **DualWriteProjectFieldConfiguration**
   - **BusinessEventsDefinition**
4. You may want to relink Finance and Operations UAT environment against Dataverse UAT environment.
5. Enable the maps.

If the backup and restore processes are running on Dataverse, then follow these steps:

1. Sign in to Finance and Operations UAT environment.
2. Stop all table maps.
3. Unlink the dual-write connection as the Dataverse UAT environment will be pointing towards Finance and Operations production environment.
4. Delete the data from the **Dual Write Runtime Configurations** table on Dataverse.
5. You may want to relink Finance and Operations UAT environment against Dataverse UAT environment.
6. Enable the maps.

**Scenario: Reset or change linking**

If you want to reset your existing sandbox Dataverse instance that is linked for dual-write or you want to change the linking to a different Dataverse instance, then follow these steps:

1. Sign in to the Finance and Operations app.
2. Stop all entity maps.
3. Unlink the dual-write connection between Finance and Operations app and Dataverse.
4. Reset the Dataverse environment.
5. Delete the data from the key tables in the Finance and Operations app.
   - DualWriteProjectConfiguration
   - DualWriteProjectFieldConfiguration
   - BusinessEventsDefinition

6. Set up dual-write on the environment that you want to reset. For more information, see System requirements and prerequisites.

Known issues

SecureConfig Organization error
When you try to copy, update, or delete records after copying the environment, the following error appears:
SecureConfig Organization (ProjOpsTest4) does not match actual CRM Organization (org6459f7a8_195867911_20200717T174709).

Follow these steps to mitigate the error:

1. In the customer engagement app, select Advanced find.
2. In the Look for field, select Dual Write Runtime Configurations.
3. Select Results.
4. Rows will be displayed. Select all the rows.
5. Select the Delete icon.
Customer data can be mastered in more than one Dynamics 365 application. For example, a customer row can originate through sales activity in Dynamics 365 Sales (a customer engagement app), or a row can originate through retail activity in Dynamics 365 Commerce (a finance and operations app). No matter where the customer data originates, it is integrated behind the scenes. Integrated customer master gives you the flexibility to master customer data in any Dynamics 365 application and provides a comprehensive view of the customer across the Dynamics 365 application suite.

Customer data flow

*Customer* is a well-defined concept in applications. Therefore, the integration of customer data just involves harmonizing the customer concept between the two applications. The following illustration shows the customer data flow.

Customers can be broadly classified into two types: commercial/organizational customers and consumers/end users. These two types of customers are stored and handled differently in Finance and Operations and Dataverse.

In Finance and Operations, both commercial/organizational customers and consumers/end users are mastered in a single table that is named `CustTable` (CustCustomerV3Entity), and they are classified based on the `Type` attribute. (If `Type` is set to `Organization`, the customer is a commercial/organizational customer, and if `Type` is set to `Person`, the customer is a consumer/end user.) The primary contact person information is handled through the SMMContactPersonEntity table.

In Dataverse, commercial/organizational customers are mastered in the Account table and are identified as customers when the `RelationshipType` attribute is set to `Customer`. Both consumers/end users and the contact person are represented by the Contact table. To provide a clear separation between a consumer/end user and a contact person, the `Contact` table has a Boolean flag that is named `Sellable`. When `Sellable` is `True`, the contact is a consumer/end user, and quotations and orders can be created for that contact. When `Sellable` is `False`, the contact is just a primary contact person of a customer.
When a non-sellable contact participates in a quotation or order process, **Sellable** is set to **True** to flag the contact as a sellable contact. A contact that has become a sellable contact remains a sellable contact.

**Templates**

Customer data includes all information about the customer, such as the customer group, addresses, contact information, payment profile, invoice profile, and loyalty status. A collection of table maps works together during customer data interaction, as shown in the following table.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APPS</th>
<th>CUSTOMER ENGAGEMENT APPS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDS Contacts V2</td>
<td>contacts</td>
<td>This template synchronizes all primary, secondary, and tertiary contact information, for both customers and vendors.</td>
</tr>
<tr>
<td>Customer groups</td>
<td>msdyn_customergroups</td>
<td>This template synchronizes customer group information.</td>
</tr>
<tr>
<td>Customer payment method</td>
<td>msdyn_customerpaymentmethods</td>
<td>This template synchronizes customer payment method information.</td>
</tr>
<tr>
<td>Customers V3</td>
<td>accounts</td>
<td>This template synchronizes customer master information for commercial and organizational customers.</td>
</tr>
<tr>
<td>Customers V3</td>
<td>contacts</td>
<td>This template synchronizes customer master data for consumers and end users.</td>
</tr>
<tr>
<td>Name affixes</td>
<td>msdyn_nameaffixes</td>
<td>This template synchronizes name affixes reference data, for both customers and vendors.</td>
</tr>
<tr>
<td>Payment day lines CDS V2</td>
<td>msdyn_paymentdaylines</td>
<td>This template synchronizes payment day lines reference data, for both customers and vendors.</td>
</tr>
<tr>
<td>Payment days CDS</td>
<td>msdyn_paymentdays</td>
<td>This template synchronizes payment days reference data, for both customers and vendors.</td>
</tr>
<tr>
<td>Payment schedule lines</td>
<td>msdyn_paymentschedulelines</td>
<td>Syncs payment schedule lines reference data, for both customers and vendors.</td>
</tr>
<tr>
<td>Payment schedule</td>
<td>msdyn_paymentschedules</td>
<td>This template synchronizes payment schedule reference data, for both customers and vendors.</td>
</tr>
<tr>
<td>Terms of payment</td>
<td>msdyn_paymentterms</td>
<td>This template synchronizes payment terms (terms of payment) reference data, for both customers and vendors.</td>
</tr>
</tbody>
</table>
The term vendor refers to a supplier organization, or a sole proprietor who supplies goods or services to a business. Although vendor is an established concept in Microsoft Dynamics 365 Supply Chain Management, no vendor concept exists in customer engagement apps. However, you can overload the Account/Contact table to store vendor information. The integrated vendor master introduces an explicit vendor concept in customer engagement apps. You can either use the new vendor design or store vendor data in the Account/Contact table. Dual-write supports both approaches.

In both approaches, the vendor data is integrated between Dynamics 365 Supply Chain Management, Dynamics 365 Sales, Dynamics 365 Field Service, and Power Apps portals. In Supply Chain Management, the data is available for workflows such as purchase requisitions and purchase orders.

**Vendor data flow**

If you don’t want to store vendor data in the Account/Contact table in Dataverse, you can use the new vendor design.

If you want to continue to store vendor data in the Account/Contact table, you can use the extended vendor design. To use the extended vendor design, you must configure the vendor workflows in the dual-write solution package. For more information, see Switch between vendor designs.
If you’re using Power Apps portals for self-service vendors, the vendor information can flow directly to Finance and Operations apps.

**Templates**

Vendor data includes all information about the vendor, such as the vendor group, addresses, contact information, payment profile, and invoice profile. A collection of table maps work together during vendor data interaction, as shown in the following table.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APPS</th>
<th>CUSTOMER ENGAGEMENT APPS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDS Contacts V2</td>
<td>contacts</td>
<td>This template synchronizes all primary, secondary, and tertiary contact information, for both customers and vendors.</td>
</tr>
<tr>
<td>Name affixes</td>
<td>msdyn_nameaffixes</td>
<td>This template synchronizes name affixes reference data, for both customers and vendors.</td>
</tr>
<tr>
<td>Payment day lines CDS V2</td>
<td>msdyn_paymentdaylines</td>
<td>This template synchronizes payment day lines reference data, for both customers and vendors.</td>
</tr>
<tr>
<td>Payment days CDS</td>
<td>msdyn_paymentdays</td>
<td>This template synchronizes payment days reference data, for both customers and vendors.</td>
</tr>
<tr>
<td>Payment schedule lines</td>
<td>msdyn_paymentschedulelines</td>
<td>Syncs payment schedule lines reference data, for both customers and vendors.</td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS APPS</td>
<td>CUSTOMER ENGAGEMENT APPS</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Payment schedule</td>
<td>msdyn_paymentschedules</td>
<td>This template synchronizes payment schedule reference data, for both customers and vendors.</td>
</tr>
<tr>
<td>Terms of payment</td>
<td>msdyn_paymentterms</td>
<td>This template synchronizes payment terms (terms of payment) reference data, for both customers and vendors.</td>
</tr>
<tr>
<td>Vendors V2</td>
<td>msdyn_vendors</td>
<td>Businesses that use a custom solution for vendors can take advantage of the out-of-box vendor concept that is being introduced in Dataverse because of Finance and Operations apps integration.</td>
</tr>
<tr>
<td>Vendor groups</td>
<td>msdyn_vendorgroups</td>
<td>This template synchronizes vendor group information.</td>
</tr>
<tr>
<td>Vendor payment method</td>
<td>msdyn_vendorpaymentmethods</td>
<td>This template synchronizes vendor payment method information.</td>
</tr>
</tbody>
</table>
Vendor data flow

If you choose to use the Account table to store vendors of the Organization type and the Contact table to store vendors of the Person type, configure the following workflows. Otherwise, this configuration isn’t required.

Use the extended vendor design for vendors of the Organization type

The Dynamics365FinanceExtended solution package contains the following workflow process templates. You will create a workflow for each template.

- Create Vendors in Accounts Table
- Create Vendors in Vendors Table
- Update Vendors in Accounts Table
- Update Vendors in Vendors Table

To create new workflow processes by using the workflow process templates, follow these steps.

1. Create a workflow process for the Vendor table, and select the Create Vendors in Accounts Table workflow process template. Then select OK. This workflow handles the vendor creation scenario for the Account table.

2. Create a workflow process for the Vendor table, and select the Update Vendors in Accounts Table workflow process template. Then select OK. This workflow handles the vendor update scenario for the
3. Create a workflow process for the **Account** table, and select the Create Vendors in Vendors Table workflow process template.

4. Create a workflow process for the **Account** table, and select the Update Vendors in Vendors Table workflow process template.

5. You can configure the workflows as either real-time workflows or background workflows, depending on your requirements. To configure a workflow as a background workflow, select *Convert to a background workflow*.

6. Activate the workflows that you created for the **Account** and **Vendor** tables to start to use the **Account** table to store information for vendors of the **Organization** type.

**Use the extended vendor design for vendors of the Person type**

The **Dynamics365FinanceExtended** solution package contains the following workflow process templates. You will create a workflow for each template.

- Create Vendors of type Person in Vendors Table
- Create Vendors of type Person in Contacts Table
- Update Vendors of type Person in Contacts Table
- Update Vendors of type Person in Vendors Table

To create new workflow processes by using the workflow process templates, follow these steps.

1. Create a workflow process for the **Vendor** table, and select the Create Vendors of type Person in Contacts Table workflow process template. Then select *OK*. This workflow handles the vendor creation scenario for the **Contact** table.

2. Create a workflow process for the **Vendor** table, and select the Update Vendors of type Person in Contacts Table workflow process template. Then select *OK*. This workflow handles the vendor update scenario for the **Contact** table.

3. Create a workflow process for the **Contact** table, and select the Create Vendors of type Person in Vendors Table template.

4. Create a workflow process for the **Contact** table, and select the Update Vendors of type Person in Vendors Table template.
5. You can configure the workflows as either real-time workflows or background workflows, depending on your requirements. To configure a workflow as a background workflow, select Convert to a background workflow.

6. Activate the workflows that you created on the Contact and Vendor tables to start to use the Contact table to store information for vendors of the Person type.
Businesses classify customers and provide sophisticated services, based on customer shopping and spending patterns. For example, Dynamics 365 Commerce has the infrastructure and functions to facilitate and handle customer loyalty cards, reward points, loyalty-based pricing, and rewards-based shopping experiences. When data about customer loyalty cards and reward points in Commerce is synced to Dataverse, customer engagement apps can use that data. For example, Dynamics 365 Customer Service users can use the data to provide the same sophisticated services through the help desk.

**Templates**

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APPS</th>
<th>CUSTOMER ENGAGEMENT APPS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loyalty card</td>
<td>msdyn_loyaltycards</td>
<td>This template syncs information about customer loyalty cards.</td>
</tr>
<tr>
<td>Loyalty levels</td>
<td>msdyn_loyaltylevels</td>
<td>This template syncs information about customer reward points.</td>
</tr>
<tr>
<td>Loyalty reward points</td>
<td>msdyn_loyaltyrewardpoints</td>
<td></td>
</tr>
</tbody>
</table>
When a business ecosystem is made up of Dynamics 365 applications, such as Finance, Supply Chain Management, and Sales, businesses often use these applications to source product data. This is because these apps provide a robust product infrastructure complemented with sophisticated pricing concepts and accurate on-hand inventory data. Businesses who use an external Product Lifecycle Management (PLM) system for sourcing the product data can channelize products from Finance and Operations apps to other Dynamics 365 apps. The unified product experience brings the integrated product data model in to Dataverse, so that all application users, including Power Platform users, can take advantage of the rich product data coming from Finance and Operations apps.

Here is the product data model from Sales.

Here is the product data model from Finance and Operations apps.
These two product data models have been integrated in Dataverse as shown below.

The dual-write table maps for products have been designed to flow data one-way only, in near-real time from Finance and Operations apps to Dataverse. However, the product infrastructure has been made open to make it bi-directional if required. Although you can customize it, it's at your own risk, as Microsoft does not recommend this approach.

**Templates**

Product information contains all the information related to the product and its definition, such as the product dimensions or the tracking and storage dimensions. As the following table shows, a collection of table maps is created to sync products and related information.
<table>
<thead>
<tr>
<th><strong>FINANCE AND OPERATIONS APPS</strong></th>
<th><strong>OTHER DYNAMICS 365 APPS</strong></th>
<th><strong>DESCRIPTION</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>All products</td>
<td>msdyn_globalproducts</td>
<td>The all products table contains all the products available in Finance and Operations apps, both the released products and the non-released products.</td>
</tr>
<tr>
<td>CDS released distinct products</td>
<td>Product</td>
<td>The <em>Product</em> table contains the columns that define the product. It includes individual products (products with subtype product) and the product variants. The following table shows the mappings.</td>
</tr>
<tr>
<td>Colors</td>
<td>msdyn_productcolors</td>
<td></td>
</tr>
<tr>
<td>Configurations</td>
<td>msdyn_productconfigurations</td>
<td></td>
</tr>
<tr>
<td>Default order settings</td>
<td>msdyn_productdefaultordersettings</td>
<td></td>
</tr>
<tr>
<td>Product categories</td>
<td>msdyn_productcategories</td>
<td>Each of the product categories and information about its structure and characteristics are contained in the product category table.</td>
</tr>
<tr>
<td>Product category assignments</td>
<td>msdyn_productcategoryassignments</td>
<td>To assign a product to a category the product category assignments table can be used.</td>
</tr>
<tr>
<td>Product category hierarchies</td>
<td>msdyn_productcategoryhierarchies</td>
<td>You use product hierarchies to categorize or group products. The category hierarchies are available in Dataverse using the Product category hierarchy table.</td>
</tr>
<tr>
<td>Product category hierarchy roles</td>
<td>msdyn_productcategoryhierarchyroles</td>
<td>Product hierarchies can be used for different roles in D365 Finance and Operations. They specify which category is used in each role the product category role table is used.</td>
</tr>
<tr>
<td>Product default order settings V2</td>
<td>msdyn_productspecificdefaultordersettings</td>
<td></td>
</tr>
<tr>
<td>Product dimension groups</td>
<td>msdyn_productdimensiongroups</td>
<td>The product dimension group defined which product dimensions define the product.</td>
</tr>
<tr>
<td>Product master colors</td>
<td>msdyn_sharedproductcolors</td>
<td>The <em>Shared product color</em> table indicates the colors that a specific product master can have. This concept is migrated to Dataverse to keep data consistent.</td>
</tr>
</tbody>
</table>
The **Shared product configuration** table indicates the configurations that a specific product master can have. This concept is migrated to Dataverse to keep data consistent.

The **Shared product size** table indicates the sizes that a specific product master can have. This concept is migrated to Dataverse to keep data consistent.

The **Shared product style** table indicates the styles that a specific product master can have. This concept is migrated to Dataverse to keep data consistent.

Product bar codes are used to uniquely identify products.

The **msdyn_sharedproductdetails** table contains the columns from Finance and Operations apps that define the product, and that contain the product's financial and management information.

The product storage dimension group represents the method used to define the placement the product in the warehouse.

The product tracking dimension group represents the method used to track the product in inventory.

Integration of products

In this model, the product is represented by the combination of two tables in Dataverse: **Product** and **msdyn_sharedproductdetails**. Whereas the first table contains the definition of a product (the unique identifier for the product, the product name, and the description), the second table contains the columns stored at the product level. The combination of these two tables is used to define the product according to the concept of the stockkeeping unit (SKU). Each released product will have its information in the mentioned tables (Product
and Shared Product Details). To keep track of all products (released and not released), the Global products table is used.

Because the product is represented as a SKU, the concepts of distinct products, product masters, and product variants can be captured in Dataverse in the following way:

- **Products with subtype product** are products that are defined by themselves. No dimensions have to be defined. An example is a specific book. For these products, one row is created in the Product table, and one row is created in the msdyn_sharedproductdetails table. No product family row is created.

- **Product masters** are used as generic products that hold the definition and rules that determine the behavior in business processes. Based on these definitions, distinct products that are known as product variants can be generated. For example, T-shirt is the product master, and it can have Color and Size as dimensions. Variants can be released that have different combinations of these dimensions, such as a small blue T-shirt or a medium green T-shirt. In the integration, one row per variant is created in the product table. This row contains the variant-specific information, such as the different dimensions. The generic information for the product is stored in the msdyn_sharedproductdetails table. (This generic information is held in the product master.) The product master information is synced to Dataverse as soon as the released product master is created (but before variants are released).

- **Distinct products** refer to all the products subtype product and all the product variants.

**Model-driven apps in Dynamics 365**

**Finance and Operations apps**

With the dual-write functionality enabled, the products from Finance and Operations will be synchronized in other Dynamics 365 products in Draft state. They are added to the first price list with the same currency used in the customer engagement app and using alphabetical sort on the price list name. In other words, they are added to the first price list in a Dynamics 365 app that matches the currency of your legal table where the product is released in a Finance and Operations app. If there is no price list for the given currency, a price list will automatically be created and the product will be assigned to it.

The current implementation of the dual-write plugins that associate the default price list to the unit look the currency associated with the Finance and Operations app and find the first price list in the customer engagement app using alphabetical sort on the price list name. To set a default price list for a specific currency
when you have multiple price lists for that currency, you must update the price list name to a name that is earlier in alphabetical order than any other price lists for that same currency. If it does not have any price list for the given currency, a new one is created.

By default products from Finance and Operations apps are synchronized to other Dynamics 365 apps in Draft state. To synchronize the product with Active state so that you can directly use it in sales order quotations, for example, the following setting needs to be chosen: System > Administration > System administration > System settings > Sales tab and select Create products in active state = yes.

When products are synchronized, you must enter a value for the Sales unit field in the Finance and Operations app, because it is a mandatory field in Sales.

The creation of product families from Dynamics 365 Sales is not supported with the dual-write synchronization of products.

The synchronization of products happens from the Finance and Operations app to Dataverse. This means that the values of the product table columns can be changed in Dataverse, but when the synchronization is triggered (when a product column is modified in a Finance and Operations app), this will overwrite the values in Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APPS</th>
<th>CUSTOMER ENGAGEMENT APPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDS released distinct products</td>
<td>Product</td>
</tr>
<tr>
<td>Released products V2</td>
<td>msdyn_sharedproductdetails</td>
</tr>
<tr>
<td>All products</td>
<td>msdyn_globalproducts</td>
</tr>
</tbody>
</table>

Product dimensions

Product dimensions are characteristics that identify a product variant. The four product dimensions (Color, Size, Style, and Configuration) are also mapped to Dataverse to define the product variants. The following illustration shows the data model for the product dimension Color. The same model is applied to Sizes, Styles and Configurations.
When a product has different product dimensions (for example, a product master has Size and Color as product dimensions), each distinct product (that is, each product variant) is defined as a combination of those product dimensions. For example, product number B0001 is an extra-small black T-shirt, and product number B0002 is a small black T-shirt. In this case, the existing combinations of product dimensions are defined. For example, the T-shirt from the preceding example can be extra-small and black, small and black, medium and black, or large and black, but it can't be extra-large and black. In other words, the product dimensions that a product master can take are specified, and variants can be released based on these values.

To keep track of the product dimensions that a product master can take, the following tables are created and mapped in Dataverse for each product dimension. For more information, see Product information overview.

### Default order settings and product-specific default order settings

Default order settings define the site and warehouse where items will be sourced from or stored, the minimum, maximum, multiple and standard quantities that will be used for trading or inventory management, the lead times, the stop flag, and the order promising method. This information is available in Dataverse using the default order settings and product-specific default order settings entity. You can read more information about the functionality in the Default order settings topic.

### Unit of measure and unit of measure conversions

The units of measure and its respective conversion are available in the Dataverse following the data model shown in the diagram.
The unit of measure concept is integrated between Finance and Operations apps and other Dynamics 365 apps. For each unit class in a Finance and Operations app, a unit group is created in a Dynamics 365 app, which contains the units belonging to the unit class. A default base unit is also created for every unit group.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APPS</th>
<th>CUSTOMER ENGAGEMENT APPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product specific unit conversions</td>
<td>msdyn_productspecificunitofmeasureconversions</td>
</tr>
<tr>
<td>Units</td>
<td>uoms</td>
</tr>
<tr>
<td>Unit conversions</td>
<td>msdyn_unitofmeasureconversions</td>
</tr>
</tbody>
</table>

**Initial synchronization of units data matching between Finance and Operations and Dataverse**

**Initial synchronization of units**

When dual write is enabled, units from Finance and Operations apps are synchronized to other Dynamics 365 apps. The unit groups synchronized from Finance and Operations apps in Dataverse have a flag set that indicates they are "Externally maintained".

**Matching units and unit classes/groups data from Finance and Operations and other Dynamics 365 apps**

First, it’s important to note that the integration key for unit is msdyn_symbol. Therefore, this value must be unique in Dataverse or other Dynamics 365 apps. Because in other Dynamics 365 apps it is the pair “Unit group ID” and “Name” that define the uniqueness of a unit, you need to consider different scenarios for matching unit data between Finance and Operations apps and Dataverse.

For units matching/overlapping in Finance and Operations apps and other Dynamics 365 apps:

- The unit belongs to a unit group in other Dynamics 365 apps that corresponds to the


associated unit class in Finance and Operations apps. In this case, the column msdyn_symbol in other Dynamics 365 apps must be filled in with the unit symbol from Finance and Operations apps. Therefore, when the data will be matched, and the unit group will be set as "Externally maintained" in other Dynamics 365 apps.

- The unit belongs to a unit group in other Dynamics 365 apps that does not correspond to the associated unit class in Finance and Operations apps (no existing unit class in Finance and Operations apps for the unit class in other Dynamics 365 apps). In this case, the msdyn_symbol must be filled in with a random string. Note that this value must be unique in other Dynamics 365 apps.

For units and unit classes in Finance and Operations not existing in other Dynamics 365 apps:

As part of dual-write the unit groups from Finance and Operations apps and its corresponding units are created and synchronized in other Dynamics 365 apps and Dataverse and the unit group will be set as "Externally maintained". No extra bootstrapping effort is required.

For units in other Dynamics 365 apps that do not exist in Finance and Operations apps:

The column msdyn_symbol must be filled in for all units. The units can always be created in Finance and Operations apps in the corresponding unit class (if it exists). If the unit class doesn’t exist, first the unit class must be created (note that you cannot create a unit class in Finance and Operations apps except through extension if you are extending the enum) matching the other Dynamics 365 apps unit group. Then you can create the unit. Note that the unit symbol in Finance and Operations apps must be the msdyn_symbol previously specified in other Dynamics 365 apps for the unit.

Product policies: dimension, tracking and storage groups

The product policies are sets of policies used for defining products and its characteristics in inventory. The product dimension group, product tracking dimension group and storage dimension group can be found as product policies.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APPS</th>
<th>CUSTOMER ENGAGEMENT APPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product dimension groups</td>
<td>msdyn_productdimensiongroups</td>
</tr>
<tr>
<td>Storage dimension groups</td>
<td>msdyn_productstoragedimensiongroups</td>
</tr>
<tr>
<td>Tracking dimension groups</td>
<td>msdyn_producttrackingdimensiongroups</td>
</tr>
</tbody>
</table>

Product hierarchies

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APPS</th>
<th>CUSTOMER ENGAGEMENT APPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product category assignments</td>
<td>msdyn_productcategoryassignments</td>
</tr>
<tr>
<td>Product category hierarchies</td>
<td>msdyn_productcategoryhierarchies</td>
</tr>
<tr>
<td>Product category hierarchy roles</td>
<td>msdyn_productcategoryhierarchyroles</td>
</tr>
</tbody>
</table>

Integration key for products

To uniquely identify products between Dynamics 365 for Finance and Operations and products in Dataverse the integration keys are used. For products, the `(productnumber)` is the unique key that identifies a product in Dataverse. It’s composed by the concatenation of: `(company, msdyn_productnumber)`. The `company`
indicates the legal entity in Finance and Operations and `msdyn_productnumber` indicates the product number for the specific product in Finance and Operations.

For users of other Dynamics 365 apps, the product is identified in the UI with the `msdyn_productnumber` (note that the label of the column is *Product number*). In the product form both the company and the `msydnn_productnumber` are shown. However, the (productnumber) column, the unique key for a product, is not shown.

If you build apps on Dataverse, you should pay attention to using the `productnumber` (the unique product ID) as the integration key. Do not use `msdyn_productnumber`, because it’s not unique.

**Initial synchronization of products and migration of data from Dataverse to Finance and Operations**

**Initial synchronization of products**

When dual-write is enabled, products from Finance and Operations apps are synchronized to Dataverse and customer engagement apps. Products created in Dataverse and other Dynamics 365 apps before dual-write was released will not be updated or matched with product data from Finance and Operations apps.

**Matching product data from Finance and Operations and other Dynamics 365 apps**

If the same products are kept (overlapping/matching) in Finance and Operations and in Dataverse and other Dynamics 365 apps, when enabling dual-write the synchronization of products from Finance and Operations will take place, and duplicate rows will appear in Dataverse for the same product. To avoid the previous situation, if other Dynamics 365 apps have products that are overlapping/matching with Finance and Operations, then the administrator enabling dual write must bootstrap the columns *Company* (example: “USMF”) and `msdyn_productnumber` (example: “1234:Black:S”) before the synchronization of products takes place. In other words, these two columns in the product in Dataverse must be filled in with the respective company in Finance and Operations to which the product needs to be matched with and with its product number.

Then, when the synchronization is enabled and takes place, the products from Finance and Operations will be synchronized with the matched products in Dataverse and other Dynamics 365 apps. This is applicable for both distinct products and product variants.

**Migration of product data from other Dynamics 365 apps to Finance and Operations**

If other Dynamics 365 apps have products that aren’t present in Finance and Operations, the administrator can first use the `EcoResReleasedProductCreationV2Entity` for importing those products in Finance and Operations. And secondly, match the product data from Finance and Operations and other Dynamics 365 apps as described above.
This topic describes the integration of site and warehouse data between Finance and Operations and Dataverse. Operational sites and warehouses are common concepts in a Supply Chain Management application. They are used to model the supply chain of your company.

**Templates**

With the integration with Dataverse, these concepts and all their related information are available in Dataverse using the sites and warehouses data tables in the following table.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APPS</th>
<th>CUSTOMER ENGAGEMENT APPS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sites</td>
<td>msdyn_operationalsites</td>
<td></td>
</tr>
<tr>
<td>Warehouses</td>
<td>msdyn_warehouses</td>
<td></td>
</tr>
</tbody>
</table>
In Finance and Operations, the concept of a *company* is both a legal construct and a business construct. It’s also a security and visibility boundary for data. Users always work in the context of a single company, and most of the data is striped by company.

Dataverse doesn’t have an equivalent concept. The closest concept is *business unit*, which is primarily a security and visibility boundary for user data. This concept doesn’t have the same legal or business implications as the company concept.

Because business unit and company aren’t equivalent concepts, it isn’t possible to force a one-to-one (1:1) mapping between them for the purpose of Dataverse integration. However, because users must, by default, be able to see the same rows in the application and Dataverse, Microsoft has introduced a new table in Dataverse that is named cdm_Company. This table is equivalent to the Company table in the application. To help guarantee that visibility of rows is equivalent between the application and Dataverse out of the box, we recommend the following setup for data in Dataverse:

- For each Finance and Operations Company row that is enabled for dual-write, an associated cdm_Company row is created.
- When a cdm_Company row is created and enabled for dual-write, a default business unit is created that has the same name. Although a default team is automatically created for that business unit, the business unit isn’t used.
- A separate owner team is created that has the same name. It’s also associated with the business unit.
- By default, the owner of any row that is created and dual-written to Dataverse is set to the “DW Owner” team that is linked to the associated business unit.

The following illustration shows an example of this data setup in Dataverse.

Because of this configuration, any row that is related to the USMF company will be owned by a team that is linked to the USMF business unit in Dataverse. Therefore, any user who has access to that business unit through a security role that is set to business unit–level visibility can now see those rows. The following example shows how teams can be used to provide the correct access to those rows.

- The “Sales Manager” role is assigned to members of the “USMF Sales” team.
- Users who have the “Sales Manager” role can access any account rows that are members of the same business unit that they are members of.
- The “USMF Sales” team is linked to the USMF business unit that was mentioned earlier.
Therefore, members of the "USMF Sales" team can see any account that is owned by the "USMF DW" user, which would have come from the USMF Company table in Finance and Operations.

As the preceding illustration shows, this 1:1 mapping between business unit, company, and team is just a starting point. In this example, a new "Europe" business unit is manually created in Dataverse as the parent for both DEMF and ESMF. This new root business unit is unrelated to dual-write. However, it can be used to give members of the "EUR Sales" team access to account data in both DEMF and ESMF by setting the data visibility to Parent/Child BU in the associated security role.

A final topic to discuss is how dual-write determines which owner team it should assign rows to. This behavior is controlled by the Default owning team column on the cdm_Company row. When a cdm_Company row is enabled for dual-write, a plug-in automatically creates the associated business unit and owner team (if it doesn’t already exist), and sets the Default owning team column. The admin can change this column to a different value. However, the admin can’t clear the column as long as the table is enabled for dual-write.

Company striping and bootstrapping

Dataverse integration brings company parity by using a company identifier to stripe data. As the following illustration shows, all company-specific tables are extended so that they have a many-to-one (N:1) relationship with the cdm_Company table.
For rows, after a company is added and saved, the value becomes read-only. Therefore, users should make sure that they select the correct company.

Only rows that have company data are eligible for dual-write between the application and Dataverse.

For existing Dataverse data, an admin-led bootstrapping experience will soon be available.

Autopopulate company name in customer engagement apps

There are several ways to auto-populate the company name in customer engagement apps.

- If you are a system administrator, you can set the default company by navigating to Advanced Settings > System > Security > Users. Open the User form, and in the Organization Information section, set the Company to default on Forms value.

- If you have Write access to the SystemUser table for the Business Unit level, then you can change the default company on any form by selecting a company from the Company drop-down menu.
Apply filtering based on the company context

To apply filtering based on the company context on your custom forms or on custom lookup columns added to the standard forms, open the form and use the Related Records Filtering section to apply the company filter. You must set this for each lookup column that requires filtering based on the underlying company on a given row. The setting is shown for Account in the following illustration.
Initialize company data

11/24/2021 • 4 minutes to read • Edit Online

**IMPORTANT**

Some or all of the functionality noted in this topic is available as part of a preview release. The content and the functionality are subject to change. For more information about preview releases, see Service update availability.

If you have an existing Microsoft Dataverse instance or Finance and Operations app instance that has business data, you might want to enable a dual-write connection against it. In this case, you must initialize the Dataverse data or Finance and Operations app data with company information before you enable dual-write. This initialization process is sometimes referred to as *bootstrapping*.

This topic includes sample scenarios that explain how to use Azure Data Factory to initialize data in Dataverse tables for dual-write. It doesn’t cover all tables, error handling scenarios, or lookups. Use this topic and template as a reference to set up your own Azure Data Factory pipeline to import data into Dataverse or update data in Dataverse.

**High-level scenario**

Consider the Customers table in a Finance and Operations app, and the Account table in Dataverse.

- Use initial write to copy reference and dependent tables, such as Company, Customer groups, and Terms of payment, from the Finance and Operations app to Dataverse.
- Use the Data management framework to export data from the Finance and Operations app in comma-separated values (CSV) format. For example, set up an export project in Data management to export customers from each company by using the DataAreaId field in the Finance and Operations app. This process is a one-time manual process.
- Use Azure Blob Storage to store the CSV files for lookup and transformation. Upload the CSV file for your Finance and Operations customers into Azure Blob Storage.
- Use Azure Data Factory to initialize data in Dataverse.

The following illustration shows the workflow.
This scenario is based on the following assumptions:

- The source data is in the Finance and Operations app.
- If an account exists in Dataverse, but it doesn’t exist in the Finance and Operations app, it won’t be initialized as part of this flow. Use DIXF or initial sync functionality based on the amount of data stored in Dataverse.
- All account records in the customer engagement apps have a natural key (account number) that matches the Finance and Operations natural key (`CustomerAccount`).
- Rows have a one-to-one (1:1) mapping across the apps.

**NOTE**

In both Finance and Operations apps and Dataverse, When a customer record is created, Party record gets created implicitly.

### Prerequisites

- **Azure subscription** – You have contributor access to an existing Azure subscription. If you don’t have an Azure subscription, create a free Azure account before you begin.
- **Azure storage account** – You have an Azure storage account. If you don’t have a storage account, follow the steps in Create an Azure storage account to create one.
- **Azure data factory** – Create an Azure Data Factory resource by following the steps in Create a data factory.
- **Finance and Operations app** – Use the Data management framework to export the data in CSV format. For more information, see Data management overview. In this template, customers are exported by using the `CustCustomerV3Entity` table.
- **Dynamics 365 Dataverse** – Use the credentials for the Dataverse admin user to initialize the data.
- **Dual-write** – Dual-write solutions are installed, and reference data is copied by using initial write.

### Deployment steps

**Set up an Azure storage account**

If you don’t have an Azure storage account, follow these steps in Create an Azure storage account to create one. In your storage account, create one container that is named `ce-data`. This container will store all data files. You can change the container in your datasets and pipelines as you require. Go to **Access keys**, and copy the
Deploy an Azure Data Factory template

1. Make a note of the name of the Azure data factory that you created.

2. Make a note of the connection string for the Azure storage account.

3. Make a note of the service URI of the Dataverse instance, and the admin user name and password.

   The following table shows the parameters that are required.

```
<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>DESCRIPTION</th>
<th>EXAMPLE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factory name</td>
<td>The name of your data factory</td>
<td>BootstrapDataverseDataADF</td>
</tr>
<tr>
<td>Bootstrap blob storage account</td>
<td>The connection string for blob storage</td>
<td>The value that you copied when you created the storage account</td>
</tr>
<tr>
<td>Linked Service_connection string</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bootstrap Dynamics 365 Linked Service_service Uri</td>
<td>The URI of the Dataverse instance</td>
<td><a href="https://contosod365.crm4.dynamics.com">https://contosod365.crm4.dynamics.com</a></td>
</tr>
<tr>
<td>Bootstrap Dynamics 365 Linked Service_properties_type Properties_username</td>
<td>The Dynamics 365 admin user's user ID</td>
<td><a href="mailto:adminservice@contoso.onmicrosoft.com">adminservice@contoso.onmicrosoft.com</a></td>
</tr>
<tr>
<td>Bootstrap Dynamics 365 Linked Service_password</td>
<td>The Dynamics 365 admin user's password</td>
<td>*******</td>
</tr>
</tbody>
</table>
```

4. Download the Azure Resource Manager (ARM) template file to your local directory.

5. In the Azure portal, go to Custom deployment.

6. Select Build your own template in the editor.

7. Select Load file, and find and select the ARM template file that you downloaded earlier. Then select Save.

8. Provide the required parameters, select Review, and then select Create.
After deployment, you will see **Pipelines**, **Datasets**, and **Data flows** sections in the list pane.
Run the process

1. In the Finance and Operations app, use the Data management framework to export data in CSV format. For more information, see Data management overview. In this template, customer data was exported from the CustCustomerV3Entity table. Set up CustCustomerV3Entity, and remove the FullPrimaryAddress field map from the mapping. Add the DataAreaId field to the CSV file. Rename the exported file 01-CustomersV3Export-Customers V3.csv, and upload it to the Azure storage account that you named ce-data.

<table>
<thead>
<tr>
<th>GZ</th>
<th>HA</th>
<th>HB</th>
<th>HC</th>
<th>HD</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAREHOUS</td>
<td>WAREHOUS</td>
<td>WAREHOUS</td>
<td>WRITEOFFF</td>
<td>DataAreaId</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>USMF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GBSI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GBSI</td>
</tr>
</tbody>
</table>

2. Download the sample customer file.

3. Run BootstrapAccountsPipeline from Azure Data Factory.
Because Dynamics 365 Finance is a financial system, *organization* is a core concept, and system setup starts with the configuration of an organization hierarchy. Business financials can then be tracked at the organization level and also at any level in the organization hierarchy.

Although Dataverse doesn't have the concept of an organization hierarchy, it does have a few loose concepts, such as total sales revenue. As part of Dataverse integration, the organization hierarchy data structure is added to Dataverse.

**Data flow**

A business ecosystem that consists of Finance and Operations apps and Dataverse will continue to have an organization hierarchy. This organization hierarchy is built on Finance and Operations apps, but it's exposed in Dataverse for informational and extensibility purposes. The following illustration shows the organization hierarchy information that is exposed in Dataverse as a one-way data flow from Finance and Operations apps to Dataverse.

Organization hierarchy table maps are available for one-way synchronization of data from Finance and Operations apps to Dataverse.

**Templates**

Product information contains all the information related to the product and its definition, such as the product dimensions or the tracking and storage dimensions. As the following table shows, a collection of table maps is created to sync products and related information.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APPS</th>
<th>CUSTOMER ENGAGEMENT APPS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Legal entities</td>
<td>cdm_companies</td>
<td>Provides bidirectional synchronization of legal entity (company) information.</td>
</tr>
<tr>
<td>Legal entities</td>
<td>msdyn_internalorganizations</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS APPS</td>
<td>CUSTOMER ENGAGEMENT APPS</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------------------------</td>
<td>---------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Operating unit</td>
<td>msdyn_internalorganizations</td>
<td></td>
</tr>
<tr>
<td>Organization hierarchy - published</td>
<td>msdyn_internalorganizationhierarchies</td>
<td>This template provides one-way synchronization of the Organization Hierarchy Published table.</td>
</tr>
<tr>
<td>Organization hierarchy purposes</td>
<td>msdyn_internalorganizationhierarchypurposes</td>
<td>This template provides one-way synchronization of the Organization Hierarchy Purpose table.</td>
</tr>
<tr>
<td>Organization hierarchy type</td>
<td>msdyn_internalorganizationhierarchytypes</td>
<td>This template provides one-way synchronization of the Organization Hierarchy Type table.</td>
</tr>
</tbody>
</table>

**Internal Organization**

Internal organization information in Dataverse comes from two tables, *Operating unit* and *Legal entities*. 
Every business works with a basic set of financial data, such as the fiscal calendar year, the currency that business is transacted in, the accounts that the money to run the business comes in to or goes out of, tax rates, and remittance. This data resides in Finance and Operations apps. However, it's exposed to Dataverse so that customer engagement apps can have a single source for finance and tax data. In this way, data is uniform across the business ecosystem.

Finance and tax data is integrated by using the following mappings:

- Integrated ledger
- Integrated tax master
In a business application, ledger data defines the core set up for how a company does business. For example, ledger data describes the fiscal year the company follows, the currencies it transacts in, and the accounts it uses. This topic describes the integration of this core financial data.

### Templates

Ledger data includes a collection of core financial table maps that work together during data interaction, as shown in the following table.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APPS</th>
<th>CUSTOMER ENGAGEMENT APPS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDS Exchange Rates</td>
<td>msdyn_currencyexchangerates</td>
<td></td>
</tr>
<tr>
<td>Chart of accounts</td>
<td>msdyn_chartofaccountses</td>
<td></td>
</tr>
<tr>
<td>Currencies</td>
<td>transactioncurrencies</td>
<td></td>
</tr>
<tr>
<td>Exchange rate currency pair</td>
<td>msdyn_currencyexchangeratepairs</td>
<td></td>
</tr>
<tr>
<td>Exchange rate type</td>
<td>msdyn_exchangeratetypes</td>
<td></td>
</tr>
<tr>
<td>Financial dimension format</td>
<td>msdyn_financialdimensionformats</td>
<td></td>
</tr>
<tr>
<td>Financial dimensions</td>
<td>msdyn_dimensionattributes</td>
<td></td>
</tr>
<tr>
<td>Fiscal calendar integration entity</td>
<td>msdyn_fiscalcalendars</td>
<td></td>
</tr>
<tr>
<td>Fiscal calendar period</td>
<td>msdyn_fiscalcalendarperiods</td>
<td></td>
</tr>
<tr>
<td>Fiscal calendar year integration entity</td>
<td>msdyn_fiscalcalendaryears</td>
<td></td>
</tr>
<tr>
<td>Ledger</td>
<td>msdyn_ledgers</td>
<td></td>
</tr>
<tr>
<td>Main account</td>
<td>msdyn_mainaccounts</td>
<td></td>
</tr>
<tr>
<td>Main account categories</td>
<td>msdyn_mainaccountcategories</td>
<td></td>
</tr>
</tbody>
</table>
Tax setup data defines the setup for both indirect taxes (VAT, GST, Sales tax) and withholding tax. It describes the tax calculation rule, tax rate, tax accounting, settlement, and other concepts.

**Templates**

Tax data includes a collection of table maps that work together during data interaction, as shown in the following table.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APPS</th>
<th>CUSTOMER ENGAGEMENT APPS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item sales tax group</td>
<td>msdyn_taxitemgroups</td>
<td></td>
</tr>
<tr>
<td>Sales tax authorities</td>
<td>msdyn_taxauthorities</td>
<td></td>
</tr>
<tr>
<td>Sales tax exempt code entity CDS</td>
<td>msdyn_taxexemptcodes</td>
<td></td>
</tr>
<tr>
<td>Sales tax groups</td>
<td>msdyn_taxgroups</td>
<td></td>
</tr>
<tr>
<td>Sales tax ledger posting groups V2</td>
<td>msdyn_taxpostinggroups</td>
<td></td>
</tr>
<tr>
<td>Withholding tax codes</td>
<td>msdyn_withholdingtaxcodes</td>
<td></td>
</tr>
<tr>
<td>Withholding tax groups</td>
<td>msdyn_withholdingtaxgroups</td>
<td></td>
</tr>
</tbody>
</table>
Microsoft Dynamics 365 Supply Chain Management provides robust procurement functionality. Dynamics 365 Field Service offers similar functionality that supports the purchasing processes that are associated with the service process. The functionality in these two apps is integrated through dual-write, and the resulting cross-functional use cases are enabled through table mappings, solution logic, views, and forms.

This integration supports purchase order creation and, in most cases, updates from both apps. However, Supply Chain Management controls pricing, addresses, and product receipt. Several powerful cross-functional use cases are enabled for organizations that use both Field Service and Supply Chain Management. These use cases enable procurements to be initiated and tracked across both systems.

The following illustration shows the tables in both systems and how they are mapped to each other. Purchase orders in Field Service reference an account row, whereas purchase orders in Supply Chain Management reference a vendor row. To resolve the integration, dual-write uses a reference to link vendor rows with account rows. For more information, see Integrated vendor master.

---

**Prerequisites**

To integrate Supply Chain Management with Field Service, you must install the following components:

- Field Service version 8.8.31.60 or later, for comprehensive purchase order integration
- Supply Chain Management version 10.0.14 or later
- Dual-write, to run the OneFSSCM solution

---

**Installation guidelines**

**Prerequisites**
- **Dual-write** – For more information, see the [Dual-write home page](#).
- **Dynamics 365 Field Service** – For more information, see [How to install Dynamics 365 Field Service](#).

When they are enabled in Microsoft Dataverse, dual-write and Field Service introduce several solution layers that extend the environment with new metadata, forms, views, and logic. These solutions can be enabled in any order, though you typically install in the order that is given here:

1. **Field Service Common** – Field Service Common is installed when Field Service is installed in the environment.
2. **Field Service (Anchor)** – Field Service (Anchor) is installed when Field Service is installed in the environment.
3. **Supply Chain Management Extended** – Supply Chain Management Extended is automatically installed when dual-write is enabled in an environment.
4. **OneFSSCM solution** – OneFSSCM is automatically installed by whichever solution (Field Service or Supply Chain Management) is installed last.
   - If Field Service is already installed in the environment, and you enable dual-write, which installs Supply Chain Management Extended, OneFSSCM is installed.
   - If Supply Chain Management Extended is already installed in the environment, and you install Field Service, OneFSSCM is installed.

**Initial synchronization**

To create new purchase orders and work with existing purchase orders, you must sync the reference data between Supply Chain Management and Dataverse. You use the initial write functionality to detect the table relationships and find the tables that you must enable for a given map.

You must sync the following tables:

- **Product templates**
  
  When you run the initial write, you get a full list of the tables that are required. Here are some examples of these templates:
  - All products
  - Released products V2
  - Dataverse released distinct products

- **Sites**

- **Warehouses**

- **Procurement categories templates**
  
  Here are some examples of these templates:
  - Procurement categories
  - Pro
  - Product category hierarchy
  - Product category assignments

- **Vendor templates, such as Vendor V2**

- **Contact person templates, such as Dataverse Contacts V2**

- **Worker templates, such as Worker**
Synchronization of the tables ensures that all documents (purchase orders and product receipts) in Supply Chain Management are available in Dataverse.

**Account and Vendor tables**

Purchase orders in Field Service rely on the Account table to track vendors. Therefore, the Dataverse tables for purchase orders use accounts to track vendors. To accommodate this key difference, the following four workflows must be activated to keep the accounts and vendors in sync:

- Create Vendors in Accounts table
- Create Vendors in Vendors table
- Update Vendors in Accounts table
- Update Vendors in Vendors table

If OneFSSCM is installed, because both Field Service and Supply Chain Management Extended are installed, these workflows are automatically activated. If Field Service isn’t installed, but you want to integrate the purchase order tables with Dataverse, you must activate these workflows. In both cases, unless you start from scratch, you might have to ensure that all vendors are created as accounts in Dataverse before you create purchase orders. Otherwise, errors might occur.

**Initial synchronization**

After all the prerequisites are in place, if you want existing purchase orders and product receipts to be available in both systems, you must do an initial synchronization of the following templates:

- Purchase Order Header V2
- CDS Purchase Order Line
- CDS Purchase Order Line soft delete
- Purchase Order Receipt
- Purchase Order Receipt Product

**Mappings with logic**

The procurement integration extends the product mapping with the following logic to ensure that the Field Service Product Type column is correctly set in the products table in Dataverse:

- If Product Type is set to Product, and Item model group, Stocked product is set to True, Field Service Product Type is set to Inventory.
- If Product Type is set to Product, and Item model group, Stocked product is set to False, Field Service Product Type is set to Non-Inventory.
- If Product Type is set to Service, Field Service Product Type is set to Service.

In addition, Dataverse includes logic that maps vendors with their related accounts. This logic sets the default invoice vendor account. On create, server-side plug-in logic sets the default invoice vendor account from the vendor that is related to the account. The vendor has a reference to the invoice account that is used to set this value.

**Supported scenarios**

- Purchase orders can be created and updated by Dataverse users. However, the process and data are controlled by Supply Chain Management. The constraints on updates to purchase order columns in Supply Chain Management apply when updates come from Field Service. For example, you can’t update a purchase order if it has been finalized.

- If the purchase order is controlled by change management in Supply Chain Management, a Field Service user can update the purchase order only when the Supply Chain Management approval status is Draft.
- Several columns are managed only by Supply Chain Management and can't be updated in Field Service. To learn which columns can't be updated, review the mapping tables in the product. For the sake of simplicity, most of these columns are set to read-only on Dataverse pages.

  For example, the columns for price information are managed by Supply Chain Management. Supply Chain Management has trade agreements that Field Service can benefit from. Columns such as **Unit price**, **Discount**, and **Net amount** come only from Supply Chain Management. To ensure that the price is synced to Field Service, you should use the **Sync** feature on the **Purchase Order** and **Purchase Order Product** pages in Dataverse when purchase order data has been entered. For more information, see **Sync with the Dynamics 365 Supply Chain Management procurement data on demand**.

- The **Totals** column is available only in Field Service, because there are no up-to-date totals of the purchase order in Supply Chain Management. The totals in Supply Chain Management are calculated based on multiple parameters that aren't available in Field Service.

- Purchase order lines where only a procurement category is specified, or where the product that is specified is an item of the **Service** product type or Field Service product type, can be initiated only in Supply Chain Management. The lines are then synced to Dataverse and are visible in Field Service.

- If only Field Service is installed, not Supply Chain Management, the **Warehouse** column is mandatory on the purchase order. However, if Supply Chain Management is installed, this requirement is relaxed, because Supply Chain Management allows for purchase order lines where no warehouse is specified in certain situations.

- Product receipts (purchase order receipts in Dataverse) are managed by Supply Chain Management and can't be created from Dataverse if Supply Chain Management is installed. The product receipts from Supply Chain Management are synced from Supply Chain Management to Dataverse.

- Under-delivery is allowed in Supply Chain Management. The OneFSSCM solution adds logic so that, when the product receipt line (or purchase order receipt product in Dataverse) is created or updated, an inventory journal row is created in Dataverse to adjust the remaining quantity that is on order for under-delivery scenarios.

**Unsupported scenarios**

- Field Service prevents lines from being added to a canceled purchase order in Supply Chain Management. As a workaround, you can change the system status of the purchase order in Field Service, and then add the new line in either Field Service or Supply Chain Management.

- Although procurement rows affect inventory levels in both systems, this integration doesn't ensure inventory alignment across Supply Chain Management and Field Service. Both Field Service and Supply Chain Management have other processes that update inventory levels. Those processes are outside the scope of procurement.

**Status management**

The statuses of purchase orders in Field Service differ from the statuses in Supply Chain Management.

**Field Service purchase order and purchase order product statuses**
Supply Chain Management purchase order and purchase order line statuses

Line approval statuses are active only when there is a line workflow.

The following rules are applied to the status columns:

- The status in Supply Chain Management can't be updated from Field Service. However, in some cases, the status in Field Service will be updated when the purchase order status in Supply Chain Management is changed.
- If a purchase order in Supply Chain Management is under change management, and a change is being processed, the approval status is Draft or In Review. In this case, the Field Service approval status will be set to Null.
- If the purchase order approval status in Supply Chain Management is set to Approved, In External review, Confirmed, or Finalized, the Field Service purchase order approval status will be set to Approved.
- If the purchase order approval status in Supply Chain Management is set to Rejected, the Field Service purchase order approval status will be set to Rejected.
- If the document header status in Supply Chain Management is changed to Open order (Back order), and the Field Service purchase order status is Draft or Cancelled, the Field Service purchase order status will be changed to Submitted.
- If the document header status in Supply Chain Management is changed to Cancelled, and no purchase order receipt products in Field Service are associated with the purchase order (via purchase order products), the Field Service system status is set to Cancelled.
- If purchase order line status in Supply Chain Management is Cancelled, the purchase order product status in Field Service is set to Cancelled. In addition, if the purchase order line status in Supply Chain Management is changed from Cancelled to Back Order, the purchase order product item status in Field Service is set to Pending.

Sync with the Supply Chain Management procurement data on demand

Supply Chain Management includes procurement data that handles trade agreements, discounts, and other scenarios that rely on secondary processes in Supply Chain Management. The procurement engine uses complex rules to determine the best price for a given purchase order. When you use dual-write, data isn't always kept synchronous across the two environments, especially in scenarios where the row was created or updated.
Sync the procurement data from Supply Chain Management

1. In Dataverse, go to Inventory > Purchase Order.
2. Select New to create a new purchase order, or select the row for an existing purchase order.
3. From the purchase order or purchase order line.
4. On the Action Pane, select Sync.

All columns from Dataverse and Field Service that are shared by Supply Chain Management are synced.

Here are the situations where you might use the Sync function:

- If you make multiple successive changes to the same row from Dataverse, run the Sync function.
- If you aren’t sure whether a change might be the second successive change from Dataverse, it might make sense to run the Sync function.
- If you receive an error message about updating a value from Supply Chain Management, run the Sync function, and then retry the update in Dataverse.

Cancelling the posting process

If the product receipt posting process is cancelled during processing, then dual-write might create a product receipt row in Dataverse, but not create a product receipt row in Supply Chain Management. This situation happens because dual-write does not support distributed transactions.

Templates

The following templates are available for the integration of procurement-related documents.

<table>
<thead>
<tr>
<th>SUPPLY CHAIN MANAGEMENT</th>
<th>FIELD SERVICE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purchase order header V2</td>
<td>msdyn_Purchaseorders</td>
<td>This table contains the columns that represent the purchase order header.</td>
</tr>
<tr>
<td>Purchase order line entity</td>
<td>msdyn_PurchaseOrderProducts</td>
<td>This table contains the rows that represent lines on a purchase order. The product number is used for synchronization. This identifies the product as a stock keeping unit (SKU), including product dimensions. For more information about product integration with Dataverse, see Unified product experience.</td>
</tr>
<tr>
<td>Product receipt header</td>
<td>msdyn_purchaseorderreceipts</td>
<td>This table contains the product receipt headers that are created when a product receipt is posted in Supply Chain Management.</td>
</tr>
<tr>
<td>Product receipt line</td>
<td>msdyn_purchaseorderreceiptproducts</td>
<td>This table contains the product receipt lines that are created when a product receipt is posted in Supply Chain Management.</td>
</tr>
<tr>
<td>SUPPLY CHAIN MANAGEMENT</td>
<td>FIELD SERVICE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>--------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Purchase order line soft deleted entity</td>
<td>msdyn_purchaseorderproducts</td>
<td>This table contains information about purchase order lines that are soft-deleted. A purchase order line in Supply Chain Management can be soft-deleted only when the purchase order has been confirmed or approved, if change management is turned on. The row exists in the Supply Chain Management database and is marked as <code>IsDeleted</code>. Because Dataverse doesn't have a concept of soft-deletion, it's important that this information be synced to Dataverse. In this way, lines that are soft-deleted in Supply Chain Management can automatically be deleted from Dataverse. In this case, the logic for deleting a line in Dataverse is located in Supply Chain Management Extended.</td>
</tr>
</tbody>
</table>
Microsoft Dynamics 365 Supply Chain Management includes a pricing engine that handles trade agreements, price lists, customer loyalty programs, promotions, and discounts. The pricing engine uses complex rules to determine the best price for a given quotation or order. When you use dual-write, you use either static pricing or the pricing engine from Dynamics 365 Supply Chain Management on the Quote and Order pages in Dynamics 365 Sales.

**Use the pricing engine from Supply Chain Management in Sales**

1. In Sales, go to **Sales > Orders**.
2. Select **New** to create a new order, or select an existing order in the **My Orders** list.
3. Add a new order line.
4. If you’re creating a new order, select **Price Order** on the Action Pane. If you’re updating an existing order, select **Recalculate** on the Action Pane.

   The following columns are automatically filled in:
   - Detail Amount
   - Discount %
   - Discount
   - Pre-Freight Amount
   - Freight Amount
   - Total Tax
   - Total Amount

5. To ensure that the system considers trade agreements to calculate the price:
   a. Navigate to your Supply Chain Management environment.
   b. Navigate to **Accounts receivable > Setup > Accounts receivable parameters**.
   c. Select the **Prices** tab in the side navigation bar.
   d. Under the **Trade agreement evaluation** fastab, uncheck the **Manual entry** option.

**How it works**

When you select **Price Order** in Sales, the **Totals** function on the **Sales Order > View** tab in Supply Chain Management is called for the associated sales order. The values in the order total in Sales are used to fill in the corresponding columns in Supply Chain Management.

When the sales order total is calculated in Supply Chain Management, the calculation evaluates the existing trade agreements for the customer and the products that are listed in the sales order. This information is used to calculate the totals. When **Price Order** is selected, Sales automatically reflects all the setup that has been done in Supply Chain Management.

**Limitations**
When the columns in Sales are filled in, the following limitations apply:

- The setup of charges and charge allocations in Supply Chain Management isn’t replicated in Sales.
- Pricing doesn’t consider special retail pricing that is specified in the Retail Channel column on the sales order line page in Supply Chain Management.
- Discounts that are defined in the Trade Allowance Management section of Supply Chain Management aren’t considered.
- Pricing doesn’t consider sales agreements.
An important goal of most businesses is to convert prospects to customers and then maintain an ongoing business relationship with those customers. In Microsoft Dynamics 365 apps, the prospect-to-cash process occurs through quotations or order processing workflows, and the financials are reconciled and recognized. Integration of prospect-to-cash with dual-write creates a workflow that takes a quotation and an order that originate in either Dynamics 365 Sales or Dynamics 365 Supply Chain Management, and makes the quotation and order available in both apps.

In the app interfaces, you can access the processing statuses and invoice information in real time. Therefore, you can more easily manage functions such as product stocking, inventory handling, and fulfillment in Supply Chain Management, without having to re-create the quotations and orders.

For information about customer and contact integration, see Integrated customer master. For information about product integration, see Unified product experience.

**NOTE**

In Dynamics 365 Sales, both prospect and customer refer to a record in the Account table where the RelationshipType column is either Prospect or Customer. If your business logic includes an Account qualification process where the Account record is created and qualified as a prospect first and then as a customer, that record synchronizes to the Finance and Operations app only when it is a customer (RelationshipType=Customer). If you want the Account row to synchronize as a prospect, then you need a custom map to integrate the prospect data.

Prerequisites and mapping setup

Before you can sync sales quotations, you must update the following settings.

**Setup in Sales**

In Sales, go to Settings > Administration > System settings > Sales, and make sure that the following settings are used:

- The Use system pricing calculation system option is set to Yes.
- The Discount calculation method column is set to Line item.

**Sites and warehouses**

In Supply Chain Management, the Site and warehouse columns are required for quotation lines and order lines. If you set the site and warehouse in the default order settings, those columns will automatically be set when you add a product to a quotation line or an order line.

**Number sequences for quotations and orders**

The number sequences for Supply Chain Management and Sales aren't connected when quotations and orders are created and synced in Sales and Supply Chain Management. If a sales order that is created in Sales is synced to Supply Chain Management, it has the same sales order number in Supply Chain Management. To help ensure that the sales order number isn't duplicated, you must use different number sequence systems in the two apps.

For example, the number sequence in Supply Chain Management is 1, 2, 3, 4, 5, ..., and the number sequence in Sales is 100, 99, 98, .... If you create 100 sales orders in Sales, an order number will eventually be generated
that already exists in Supply Chain Management. In other words, the two number sequences will eventually overlap as sales orders are created in Supply Chain Management and Sales. Instead, you might use a number sequence such as F1, F2, F3, ... in Supply Chain Management and a number sequence such as C1, C2, C3, ... in Sales. These number sequences will never produce duplicate sales order numbers.

Sales quotations

Sales quotations can be created in either Sales or Supply Chain Management. If you create a quotation in Sales, it’s synced to Supply Chain Management in real time. Likewise, if you create a quotation in Supply Chain Management, it’s synced to Sales in real time. Note the following points:

- You can add a discount to the product on the quotation. In this case, the discount will be synced to Supply Chain Management. The Discount, Charges, and Tax columns on the header are controlled by a setup in Supply Chain Management. This setup doesn’t support integration mapping. Instead, the Price, Discount, Charge, and Tax columns are maintained and handled in Supply Chain Management.
- The Discount %, Discount, and Freight Amount columns on the sales quotation header are read-only columns.
- The Freight terms, Delivery terms, Shipping method, and Delivery mode columns aren't part of the default mappings. To map these columns, you must set up a value mapping that is specific to the data in the organizations that the table is synced between.

If you are also using the Field Service solution, make sure to re-enable the Quote Line Quick Create parameter. Re-enabling the parameter lets you continue creating quote lines using the quick create function.

1. Navigate to your Dynamics 365 Sales application.
2. Select the settings icon in the top navigation bar.
3. Select Advanced Settings.
4. Choose the Customize the System option.
5. Select the Quote Line menu item.
6. Go to the Data Services section and select the Allow quick create checkbox.

Sales orders

Sales orders can be created in either Sales or Supply Chain Management. If you create a sales order in Sales, it’s synced to Supply Chain Management in real time. Likewise, if you create a sales order in Supply Chain Management, it’s synced to Sales in real time. Note the following points:

- Write-in products on Dynamics 365 Sales will appear as product categories in Dynamics 365 Supply Chain Management.
- Discount calculation and rounding:
  - The discount calculation model in Sales differs from the discount calculation model in Supply Chain Management. In Supply Chain Management, the final discount amount on a sales line can be the result of a combination of discount amounts and discount percentages. If this final discount amount is divided by the quantity on the line, rounding can occur. However, this rounding isn’t considered if a rounded per-unit discount amount is synced to Sales. To help ensure that the full discount amount from a sales line in Supply Chain Management is correctly synced to Sales, the full amount must be synced without being divided by the line quantity. Therefore, you must define the discount calculation method as Line item in Sales.
  - When a sales order line is synced from Sales to Supply Chain Management, the full line discount amount is used. Because Supply Chain Management has no column that can store the full discount amount for a line, the amount is divided by the quantity and stored in the Line discount column. Any rounding that occurs during this division is stored in the Sales charges column on the sales line.
Example: Synchronization from Sales to Supply Chain Management

You have the following sales order:

- **Sales**: Quantity = 3, per-line discount = $10.00
- **Supply Chain Management**: Quantity = 3, line discount amount = $3.33, sales charge = –$0.01

If you sync from Supply Chain Management to Sales, you get the following result:

- **Supply Chain Management**: Quantity = 3, line discount amount = $3.33, sales charge = –$0.01
- **Sales**: Quantity = 3, per-line discount = (3 × $3.33) + $0.01 = $10.00

Dual-write solution for Sales

New columns have been added to the **Order** table and appear on the page. Most of these columns appear on the **Integration** tab in Sales. To learn more about how the status columns are mapped, see [Set up the mapping for sales order status columns](#).

- The **Create Invoice** and **Cancel Order** buttons on the **Sales order** page are hidden in Sales.
- The **Sales order status** value will remain **Active** to help ensure that changes from Supply Chain Management can flow to the sales order in Sales. To control this behavior, set the default **Statecode [Status]** value to **Active**.

Invoices

Sales invoices are created in Supply Chain Management and synced to Sales. Note the following points:

- An **Invoice number** column has been added to the **Invoice** table and appears on the page.
- The **Create invoice** button on the **Sales order** page is hidden, because invoices will be created in Supply Chain Management and synced to Sales. The **Invoice** page can’t be edited, because invoices will be synced from Supply Chain Management.
- The **Sales order status** value is automatically changed to **Invoiced** when the related invoice from Supply Chain Management has been synced to Sales. Additionally, the owner of the sales order that the invoice was created from is assigned as the owner of the invoice. Therefore, the owner of the sales order can view the invoice.
- The **Freight terms**, **Delivery terms**, and **Delivery mode** columns aren’t included in the default mappings. To map these columns, you must set up a value mapping that is specific to the data in the organizations that the table is synced between.

Templates

Prospect-to-cash includes a collection of core table maps that work together during data interaction, as shown in the following table.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APPS</th>
<th>CUSTOMER ENGAGEMENT APPS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>All products</td>
<td>msdyn_globalproducts</td>
<td></td>
</tr>
<tr>
<td>Customers V3</td>
<td>accounts</td>
<td></td>
</tr>
<tr>
<td>Customers V3</td>
<td>contacts</td>
<td></td>
</tr>
<tr>
<td>Contacts V2</td>
<td>msdyn_contactforparties</td>
<td></td>
</tr>
<tr>
<td>CDS sales order headers</td>
<td>salesorders</td>
<td></td>
</tr>
</tbody>
</table>
For information about price lists, see [Unified product experience](#).

**Limitations**

- Return orders are not supported.
- Credit notes are not supported.
- Financial dimensions must be set for the master data, for example, customer and vendor. When a customer is added to a quotation or sales order, the financial dimensions associated with the customer record flow to the order automatically. Currently dual-write does not include financial dimensions data for master data.
This topic describes how to use the Microsoft Dynamics 365 Commerce pricing engine to create sales quotations in Dynamics 365 Sales.

The Dynamics 365 Commerce pricing engine supports most business-to-consumer (B2C) pricing scenarios, such as store-level pricing, affiliation-based and loyalty-based pricing, mix-and-match discounts, quantity discounts, and threshold discounts. The pricing engine uses complex rules to determine the best price for a given quotation or order.

When you use dual-write, you have three options for your pricing needs. You can use the static pricing that comes from the price list in Dynamics 365 Sales, the pricing engine in Dynamics 365 Supply Chain Management, or the pricing engine in Dynamics 365 Commerce. Among these options, the Commerce pricing engine is best suited to B2C scenarios.

**Use the Commerce pricing engine in Sales**

**NOTE**

Currently, the Commerce pricing engine can be used only for quotation creation in the Sales. Sales order integration with the Commerce pricing engine isn't yet available.

When users initiate a quotation in Sales, the dual-write framework copies the quotation details to Commerce. Any changes to existing quotation lines or any newly added quotation lines in Sales are copied to Commerce. When users want to use the Commerce pricing engine to price the quotation, they can select **Price quote** to update the prices of the quotation, based on the Commerce pricing engine. Prices are then automatically updated in both Sales and Commerce.

**Prerequisites**

- Before you can use the Commerce pricing engine in Sales, you must follow the steps in Prospect-to-cash in dual-write.
- You must turn off trade agreement evaluation for manual entry by following these steps:
  1. In your Commerce environment, go to Accounts receivable > Setup > Accounts receivable parameters.
  2. On the Prices tab, on the Trade agreement evaluation FastTab, clear the Manual entry check box.

**Additional resources**

Prospect-to-cash in dual-write
Microsoft Dynamics 365 Field Service is designed to service customer assets. Asset management for Dynamics 365 Supply Chain Management is designed to maintain in-house assets. Integration of these two apps lets you use Field Service to service both customer assets and in-house assets. You can also classify the assets, based on functional location or hierarchy, and track the servicing at a detailed level.

For more information, see [Integrate Dynamics 365 Field Service and Supply Chain Management](#).

### Templates

In-house assets include a collection of core table maps that work together during data interaction, as shown in the following table.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APPS</th>
<th>CUSTOMER ENGAGEMENT APPS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asset management asset lifecycle models</td>
<td>msdyn_assetlifecyclemodels</td>
<td></td>
</tr>
<tr>
<td>Asset management asset lifecycle states</td>
<td>msdyn_assetlifecyclestates</td>
<td></td>
</tr>
<tr>
<td>Asset management asset types</td>
<td>msdyn_customerassetcategories</td>
<td></td>
</tr>
<tr>
<td>Asset management assets</td>
<td>msdyn_customerassets</td>
<td></td>
</tr>
<tr>
<td>Asset management functional location lifecycle models</td>
<td>msdyn_functionallocationlifecyclemodels</td>
<td></td>
</tr>
<tr>
<td>Asset management functional location lifecycle states</td>
<td>msdyn_functionallocationlifecyclestates</td>
<td></td>
</tr>
<tr>
<td>Asset management functional location types</td>
<td>msdyn_functionallocationtypes</td>
<td></td>
</tr>
<tr>
<td>Asset management functional locations</td>
<td>msdyn_functionallocations</td>
<td></td>
</tr>
<tr>
<td>Asset management manufacturers</td>
<td>msdyn_manufacturers</td>
<td></td>
</tr>
<tr>
<td>Asset management models</td>
<td>msdyn_models</td>
<td></td>
</tr>
<tr>
<td>Asset management warranty</td>
<td>msdyn_warranties</td>
<td></td>
</tr>
</tbody>
</table>
By using inventory availability, you can check your inventory before you add a product to the Quotations, Orders, or Invoices page in Microsoft Dynamics 365 Sales. For example, you check inventory and determine a fulfillment date as one key task in the prospect-to-cash process.

If you don't have enough inventory, you can estimate a delivery date, based on projected inventory receipts and issues. You can also check the product's available-to-promise (ATP) information, where you can find the ATP quantity in the predefined time fence.

**On-hand inventory**

In Dynamics 365 Sales, a new **On-hand Inventory** button has been added to the header of the Quotes, Orders, and Invoices pages. When you select this button, a dialog box appears, where you can specify the company and the product that you want to check the on-hand inventory for. This dialog box shows the same information as **On-hand inventory**.

The dialog box returns the inventory information from Dynamics 365 Supply Chain Management. This information includes the following quantities:

- On-hand quantity
- Reserved on-hand quantity
- Available on-hand quantity
- Ordered quantity
- On-order quantity
- Reserved ordered quantity
- Total available quantity

**ATP information**

In Sales, a new **ATP Information** button has been added to line items on the Quotes, Orders, and Invoices pages. When you select this button, a dialog box appears, where you can specify the company, product, inventory site, inventory warehouse, and order quantity. This dialog box has the same settings that are described in **Order promising**.

The dialog box returns the ATP information from Supply Chain Management. This information includes the following quantities:

- ATP quantity
- Receipt quantity
- Issue quantity
- On-hand quantity

**How it works**

When you select the **On-hand Inventory** button on the Quotes, Orders, or Invoices page, a live dual-write call is made to the **Onhand inventory** API. The API calculates the on-hand inventory for the given product. The result is stored in the **InventCDSInventoryOnHandRequestEntity** and **InventCDSInventoryOnHandEntryEntity** tables, and then is written to Dataverse by dual-write. To use this
Functionality, you need to run the following dual-write maps. Skip initial synchronization when you run the maps.

- CDS inventory on-hand entries (msdyn_inventoryonhandentries)
- CDS inventory on-hand requests (msdyn_inventoryonhandrequests)

## Templates

The following templates are available for the exposing the onhand inventory data.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APPS</th>
<th>CUSTOMER ENGAGEMENT APPS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDS inventory on-hand entries</td>
<td>msdyn_inventoryonhandentries</td>
<td></td>
</tr>
<tr>
<td>CDS inventory on-hand requests</td>
<td>msdyn_inventoryonhandrequests</td>
<td></td>
</tr>
</tbody>
</table>
While mastered in only one app, worker data can be synchronized across multiple Dynamics 365 apps. For example, human resources (HR) data can be mastered in Dynamics 365 Human Resources and synchronized with Dynamics 365 Commerce, Dynamics 365 Finance, and Dynamics 365 Supply Chain Management. The data is integrated seamlessly behind the scenes. The ability to integrate data about workers ensures you're working with the same data across all Dynamics 365 apps, providing you a comprehensive view of your information.

Human resources

The integration of HR data involves just mapping the HR data between finance and operations apps and customer engagement apps.

Templates

HR data includes information about employees and contractors, positions, and jobs. A collection of table maps works together during data interaction, as shown in the following table.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APPS</th>
<th>CUSTOMER ENGAGEMENT APPS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compensation job function</td>
<td>cdm_jobfunctions</td>
<td></td>
</tr>
<tr>
<td>Compensation job type</td>
<td>cdm_jobtypes</td>
<td></td>
</tr>
<tr>
<td>Employment job functions</td>
<td>msdyn_employmentjobfunctions</td>
<td></td>
</tr>
<tr>
<td>Employment per company</td>
<td>cdm_employments</td>
<td></td>
</tr>
<tr>
<td>Jobs</td>
<td>cdm_jobs</td>
<td></td>
</tr>
<tr>
<td>Positions V2</td>
<td>cdm_jobpositions</td>
<td></td>
</tr>
<tr>
<td>Position type</td>
<td>cdm_positiontypes</td>
<td></td>
</tr>
<tr>
<td>Position worker assignments</td>
<td>cdm_positionworkerassignmentmaps</td>
<td></td>
</tr>
<tr>
<td>Worker</td>
<td>cdm_workers</td>
<td>In Dynamics 365 Finance and Supply Chain Management data, workers are classified as either employees or contractors. Dataverse can also classify workers as volunteers. Volunteers will become contractors when the data is transformed back into Finance and Supply Chain Management.</td>
</tr>
</tbody>
</table>
Party and global address book are concepts in Finance and Operations applications. A party can be an organization or a person. It's convenient to globally store and manage properties of a party, such as the name, language, contacts, and addresses. Then, when a property value is changed in one place, the change is reflected in all places where the party is involved.

Party

A party is a person or an organization that is involved in a business. When the party concept is used, a person or an organization can play more than one role in a business (for example, worker, customer, vendor, or contact). The role is based on the context and purpose. Here are some examples of roles from two fictitious companies, Contoso and Fabrikam:

- **Worker** – An employee. An example is an employee of Contoso.
- **Vendor** – A supplier organization, or a sole proprietor who supplies goods or services to a business. For example, if Fabrikam sells supplies to Contoso, Fabrikam is a vendor of Contoso.
- **Contact** – A person to contact. For example, if Contoso buys supplies from Fabrikam, employees at Contoso will reach out to the contact at Fabrikam.
- **Customer** – A person or company that buys things from a company. For example, if Contoso buys supplies from Fabrikam, Contoso is a customer of Fabrikam.

The party model is often used to represent medium to complex relationships between organizations and people, especially when a party plays more than one role. Here are some common examples:

- A party can be both a customer and a vendor. For example, in North America, Fabrikam sells electric wires to Contoso and buys assembled speakers from Contoso. In Europe, Fabrikam sells parts to Contoso, but it doesn't buy anything from Contoso.
- A party can be both an employee and a customer. For example, an employee of Contoso buys electronics from Contoso for personal use.
- There can be a many-to-many (N:N) relationship between a person and an organization. For example, Fabrikam provides service specialists and employs a placement coordinator. The placement coordinator matches service specialists to work requests from several of Fabrikam’s customers. Contoso is one of Fabrikam's customers. When Contoso requires a service specialist, it contacts the placement coordinator, who then facilitates the request. Because the placement coordinator handles requests for all customers, an N:N relationship is involved.

The following illustration shows the data model for party.
TIP

When you’re trying to create a new account record, use the **Party** field to search for the record by name. In this way, if you find the record, you just have to select it. The system then automatically fills in all the data from the party. You don’t have to manually set all the required fields. This behavior can be found on the out-of-box **Account**, **Contact**, and **Vendor** pages.

Dual-write doesn’t support all party roles of Finance and Operations apps. For a complete list of party roles, see Global address book overview.

Global address book

The global address book is a directory of postal and electronic addresses of the organizations and individuals that participate in a business.

The global address book stores and handles as many postal addresses and electronic addresses as required. For example, Fabrikam has gas stations in 50 locations. Each location has a different postal address, email address, and phone number. All business purchases are billed to the main gas station but shipped directly to the specific gas station that requested the purchase. The global address book stores the main gas station as the billing address for Fabrikam and stores each gas station as a shipping address. The addresses can be stored one time...
and then retrieved as they are required for quotations and orders.

Depending on the business context, a person or an organization might play more than one role, and the same postal address and electronic address might be used for all the roles. In this case, a change of address in one role should appear in all the other roles. The global address book stores and handles addresses globally.

The following illustration shows the data model for the global address book.

**Contact**

In customer engagement apps, a contact is a person. However, the Contact table has been overloaded to represent a person, a portal user, a business-to-consumer (B2C) customer, or a vendor. The representation is implicit, and you can’t tell the difference unless you examine related transactions. The Contact table has been limited to a one-to-one (1:1) relationship with the Account table. As part of the party and global address book model, dual-write introduces explicit properties for classification and allows for N:N relationships between a contact that is a person and an organization (Account or Vendor entity).

There are two types of Contact rows:

- **Striped contact** – A Contact row where the Company field has a mandatory value.
- **Unstriped contact** – A Contact row where the Company field is blank.

The Contact table can store the following types of rows.

<table>
<thead>
<tr>
<th>ROW TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>A person who is a customer (for example, a sellable contact or a B2C customer)</td>
<td>A striped contact record where the Company field isn’t blank and the Is Customer field is set to Yes.</td>
</tr>
<tr>
<td>A person who is a vendor (for example, a sole proprietor such as a vendor)</td>
<td>A striped contact record where the Company field isn’t blank and the Is Vendor field is set to Yes.</td>
</tr>
</tbody>
</table>
Contact for Party table

The Contact for Party table stores and handles N:N relationships between Account rows and Contact rows. It can filter out the striped Contact rows from unstriped rows and associate only the unstriped Contact rows with Account or Vendor rows.

For example, Natasha Jones and Miguel Reyes are veterinarians who provide care for farms in their areas. Natasha serves the Seattle area, and Miguel serves the Kent area. In the customer engagement app, the farms are represented as customers, and the veterinarians are represented as contact persons. A single Contact record for Natasha is associated with all the farms that Natasha works with. Likewise, a single Contact record for Miguel is associated with all the farms that Miguel works with.

These relationships are stored in the Contact for Party table. You can find the information on the out-of-box Account, Contact, and Vendor pages:

- On the Account page, you can use the Associated Contacts tab to associate one or more contacts with the Account row. In this way, you assign contact persons for an organization. You can then select one contact as the primary contact for the account. If you use the Quick create page, you can only select a contact person. The behavior is the same when you're using the Vendor page and the record type is Organization.

- On the Contact page, when the row is a customer, a vendor, or both (a striped contact), you can use the Associated Contacts tab to associate one or more contacts. In this way, you assign contact persons for a B2C customer or vendor. You can then select one contact as the primary contact. If you use the Quick create page, you can only select a contact person.

- On the Contact page, when the row is a contact person (an unstriped contact), you can use the Associated Organizations tab to associate one or more customers or vendors. In this way, you assign customers or vendors to the underlying contact person. The customer or vendor can be an organization, a person, or both. You can select a value in only one of the four fields at a time:
  - If you select a value in the Party ID field, the underlying contact is assigned to all the roles of the selected party.
  - If you select a value in the Associated Contact field, you're selecting the striped contact of the Person type.
  - If you select a value in the Associated Account or Associated Vendor field, you're selecting an organization.

<table>
<thead>
<tr>
<th>ROW TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>A person who is both a customer and a vendor</td>
<td>An striped contact record where the Company field isn't blank, the Is Customer field is set to Yes, and the Is Vendor field is set to Yes. A person can be both a producer for one product and a consumer for another product. Both Finance and Operations apps and dual-write support this relationship.</td>
</tr>
<tr>
<td>A person who is a contact person for an organization, but isn't a customer or a vendor</td>
<td>An unstriped contact record where the Company field is blank, the Is Customer field is set to No, and the Is Vendor field is set to No.</td>
</tr>
</tbody>
</table>
Regardless of your selection, the association is created at the party level, it applies to all the roles of the party, and it’s stored in the **Contact for Party** entity.

**NOTE**

The display name for the **Contact for Party** table in customer engagement apps is **Contact for Customer/Vendor**.

When you open a **Contact** row where both the **Is Customer** field and the **Is Vendor** are set to **No**, the **Associated Organizations** tab is shown. Use this tab to associate one or more customer or vendor organizations with the contact.

When you open a **Contact** row where either the **Is Customer** field or the **Is Vendor** field is set to **Yes**, the **Associated Contacts** tab is shown. Use this tab to associate one or more contacts.

### Postal addresses

A new **Addresses** tab has been introduced on the **Account**, **Contact**, and **Vendor** pages. This tab supports multiple postal addresses by using a grid, as shown in the following illustration.

The grid includes the following columns:

- **Postal Address Roles** – The purpose of the postal address.
- **Is Primary** – A value that indicates whether the address is the primary address.
- **Address Number** – The address order.

You can use the **New Address** button above the grid to create as many postal addresses as you want.

The **Address 1** and **Address 2** fields on the **Summary** tab of the **Account** page correspond to the **Delivery** and **Invoice** addresses, respectively.
The Address 1, Address 2, and Address 3 fields on the Summary tab of the Contact page correspond to the Business, Delivery, and Invoice addresses, respectively.

Electronic addresses

A new Electronic Addresses tab has been introduced on the Account, Contact, and Vendor pages. This tab supports multiple electronic addresses by using a grid, as shown in the following illustration.

The grid includes the following columns:

- **Type** – The type of electronic address.
- **Is Primary** A value that indicates whether the address is the primary address.
- **Purpose** – The purpose of the electronic address.

You can use the New Electronic Address button above the grid to create as many addresses as you want.

Electronic addresses are available only in this grid. In future releases, all postal address and electronic address fields will be removed from other tabs, for example, the Summary and Details tabs. Contact details displayed on the Details tab are read-only copies of the primary electronic address, like primary phone, primary email, primary telephone, primary fax, and primary Twitter ID. During the lead qualification process you can provide both a business phone number and a mobile phone number. The business phone number is considered the primary phone if IsMobile=No and the mobile phone number is considered the secondary phone if IsMobile=Yes.

<table>
<thead>
<tr>
<th>Tip</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Addresses and Electronic Addresses tabs on the Account and Contact forms to manage postal and electronic addresses. This ensures that address data synchronizes to Finance and Operations apps.</td>
</tr>
</tbody>
</table>

Setup

1. Open your customer engagement app environment.
2. Install the latest version (2.2.2.60 or later) of Dual-write application orchestration solution.
3. Install Dual-write Party and Global Address Book Solutions.
4. Open the finance and operations app. Navigate to the Data Management module and select the Dual-write tab. The dual-write administration page opens.
5. Apply both the solutions installed in steps 2 and 3 using the Apply Solution function.
6. Stop the following maps, because they aren’t required anymore. Instead, run the Contacts V2 (msdyn_contactforparties) map.
   - CDS Contacts V2 and Contacts (refers to customer contacts)
   - CDS Contacts V2 and Contacts (refers to vendor contacts)
7. The following entity mappings are updated for party functionality, so the latest version must be applied to these mappings.
<table>
<thead>
<tr>
<th>MAP</th>
<th>UPDATE TO THIS VERSION</th>
<th>CHANGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDS Parties (msdyn_parties)</td>
<td>1.0.0.0</td>
<td>This is a new map added as part of this release.</td>
</tr>
<tr>
<td>Contacts V2 (msdyn_contactforparties)</td>
<td>1.0.0.5</td>
<td>This is a new map added as part of this release.</td>
</tr>
<tr>
<td>Customers V3 (accounts)</td>
<td>1.0.0.5</td>
<td>Removed <code>PartyNumber</code> and other party-related fields like name, personal details, postal address fields, and electronic contact address.</td>
</tr>
<tr>
<td>Customer V3 (contacts)</td>
<td>1.0.0.5</td>
<td>Removed <code>PartyNumber</code> and other party-related fields like name, personal details, postal address fields, and electronic contact address.</td>
</tr>
<tr>
<td>Vendors V2 (msdyn_vendors)</td>
<td>1.0.0.6</td>
<td>Removed <code>PartyNumber</code> and other party-related fields like name, personal details, postal address fields, and electronic contact address.</td>
</tr>
<tr>
<td>CDS Sales quotation headers (quotes)</td>
<td>1.0.0.7</td>
<td>Replaced the contact person with <code>ContactforParty</code> reference.</td>
</tr>
<tr>
<td>Sales invoice headers V2 (invoices)</td>
<td>1.0.0.4</td>
<td>Replaced the contact person with <code>ContactforParty</code> reference.</td>
</tr>
<tr>
<td>CDS Sales order headers (salesorders)</td>
<td>1.0.0.5</td>
<td>Replaced the contact person with <code>ContactforParty</code> reference.</td>
</tr>
<tr>
<td>CDS Party postal address locations (msdyn_partypostaladdresses)</td>
<td>1.0.0.1</td>
<td>This is a new map added as part of this release.</td>
</tr>
<tr>
<td>CDS postal address history V2 (msdyn_postaladdresses)</td>
<td>1.0.0.1</td>
<td>This is a new map added as part of this release.</td>
</tr>
<tr>
<td>CDS postal address locations (msdyn_postaladdresselectronics)</td>
<td>1.0.0.0</td>
<td>This is a new map added as part of this release.</td>
</tr>
<tr>
<td>Party Contacts V3 (msdyn_partyelectronicaddresses)</td>
<td>1.0.0.0</td>
<td>This is a new map added as part of this release.</td>
</tr>
<tr>
<td>Complimentary Closings (msdyn_complimentaryclosings)</td>
<td>1.0.0.0</td>
<td>This is a new map added as part of this release.</td>
</tr>
<tr>
<td>Decision making roles (msdyn_decisionmakingroles)</td>
<td>1.0.0.0</td>
<td>This is a new map added as part of this release.</td>
</tr>
<tr>
<td>Loyalty levels (msdyn_loyaltylevels)</td>
<td>1.0.0.0</td>
<td>This is a new map added as part of this release.</td>
</tr>
</tbody>
</table>
8. Before running the above maps, you must update the integration keys manually as described in the following steps. Then select **Save**.

<table>
<thead>
<tr>
<th>MAP</th>
<th>KEYS</th>
</tr>
</thead>
</table>
| Account | accountnumber [Account Number]  
msdyn_company.cdm_companycode [Company (Company Code)] |
| Contact | msdyn_contactpersonid [Account Number/Contact Person ID]  
msdyn_company.cdm_companycode [Company (Company Code)] |
| Contact For Customer/Vendor | msdyn_contactforpartynumber [Contact For Party Number]  
msdyn_associatedcompanyid.cdm_companycode [Associated Company (Company Code)] |
| Vendor | msdyn_vendoraccountnumber [Vendor Account Number]  
msdyn_company.cdm_companycode [Company (Company Code)] |

9. In Dataverse, the duplicate detection rules character limits have increased from 450 to 700 characters. This limit lets you add one or more keys to the duplicate detection rules. Expand the duplicate detection rule for the **Accounts** table by setting the following fields.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Accounts with the same account name.</td>
</tr>
<tr>
<td>Description</td>
<td>Detects account records that have the same value in the Account Name attribute.</td>
</tr>
<tr>
<td>Base Record Type</td>
<td>Account</td>
</tr>
<tr>
<td>Matching Record Type</td>
<td>Account</td>
</tr>
<tr>
<td>Account Name (field)</td>
<td>Exact Match</td>
</tr>
</tbody>
</table>
10. Expand the duplicate detection rule for the **Contacts** table by setting the following fields.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Contacts with the same first name and last name.</td>
</tr>
<tr>
<td>Description</td>
<td>Detects contact records that have the same values in the First Name and Last Name fields.</td>
</tr>
<tr>
<td>Base Record Type</td>
<td>Contact</td>
</tr>
<tr>
<td>Matching Record Type</td>
<td>Contact</td>
</tr>
<tr>
<td>First Name (field)</td>
<td>Exact Match</td>
</tr>
<tr>
<td>Last Name (field)</td>
<td>Exact Match</td>
</tr>
<tr>
<td>Company (field)</td>
<td>Exact Match</td>
</tr>
<tr>
<td>Party Id (field)</td>
<td>Exact Match</td>
</tr>
<tr>
<td>Select (field)</td>
<td>(blank)</td>
</tr>
</tbody>
</table>
11. If you are an existing dual-write user, follow the instructions in Upgrade to the party and global address book model and upgrade your data.

12. Run the maps in the following order. If you get an error that states "Project validation failed. Missing destination field...", then open the map and select Refresh Tables. Then run the map.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APP</th>
<th>CUSTOMER ENGAGEMENT APP</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDS Parties</td>
<td>msdyn_parties</td>
</tr>
<tr>
<td>CDS postal address locations</td>
<td>msdyn_postaladdresscollections</td>
</tr>
<tr>
<td>CDS postal address history V2</td>
<td>msdyn_postaladdresses</td>
</tr>
<tr>
<td>CDS Party postal address locations</td>
<td>msdyn_partypostaladdresses</td>
</tr>
<tr>
<td>Party contacts V3</td>
<td>msdyn_partyelectronicaddresses</td>
</tr>
<tr>
<td>Customers V3</td>
<td>accounts</td>
</tr>
<tr>
<td>Customers V3</td>
<td>contacts</td>
</tr>
<tr>
<td>Vendors V2</td>
<td>msdyn_vendors</td>
</tr>
<tr>
<td>Contact person titles</td>
<td>msdyn_salescontactpersontitles</td>
</tr>
<tr>
<td>Complimentary closings</td>
<td>msdyn_complimentaryclosings</td>
</tr>
<tr>
<td>Salutations</td>
<td>msdyn_salutations</td>
</tr>
<tr>
<td>Decision making roles</td>
<td>msdyn_decisionmakingroles</td>
</tr>
<tr>
<td>Employment job functions</td>
<td>msdyn_employmentjobfunctions</td>
</tr>
<tr>
<td>Loyalty levels</td>
<td>msdyn_loyaltylevels</td>
</tr>
</tbody>
</table>
The CDS Contacts V2 (contacts) map is the map that you stopped in the step 1. When you try to run other maps, these 2 maps may appear in the list of dependents. Don't run these maps.

If the party and global address book solution is installed, you must disable the plugin named Microsoft.Dynamics.SCMExtended.Plugins.Plugins.LeadPrimaryContactPostCreate: QualifyLead of lead. If you uninstall the party and global address book solution, then you must re-enable the plugin.

The msdyn_*partynumber field (a single line text field) that is included in the Account, Contact and Vendor tables should not be used going forward. The label name has a prefix of (Deprecated) for clarity. Instead, use the msdyn_partyid field. The field is a lookup to the msdyn_party table.

<table>
<thead>
<tr>
<th>TABLE NAME</th>
<th>OLD FIELD</th>
<th>NEW FIELD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account</td>
<td>msdyn_partynumber</td>
<td>msdyn_partyid</td>
</tr>
<tr>
<td>Contact</td>
<td>msdyn_partynumber</td>
<td>msdyn_partyid</td>
</tr>
<tr>
<td>msdyn_vendor</td>
<td>msdyn_vendorpartynumber</td>
<td>msdyn_partyid</td>
</tr>
</tbody>
</table>

### Templates

A collection of table maps work together for party and global address book interaction, as shown in the following table.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APP</th>
<th>CUSTOMER ENGAGEMENT APP</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contact person titles</td>
<td>msdyn_salescontactpersontitles</td>
<td></td>
</tr>
<tr>
<td>Customers V3</td>
<td>accounts</td>
<td></td>
</tr>
<tr>
<td>Customers V3</td>
<td>contacts</td>
<td></td>
</tr>
<tr>
<td>CDS Parties</td>
<td>msdyn_parties</td>
<td></td>
</tr>
<tr>
<td>CDS Party postal address locations</td>
<td>msdyn_partypostaladdresses</td>
<td></td>
</tr>
<tr>
<td>CDS postal address history V2</td>
<td>msdyn_postaladdresses</td>
<td></td>
</tr>
</tbody>
</table>
### Known issues and limitations

- In Finance and Operations apps, when you create a customer along with address and save it, the address might not synchronize to the **Address** table. This is because of a dual-write platform sequencing issue. As a workaround, create the customer first and save it. Then add the address.

- In Finance and Operations apps, when a customer record has a primary address and you create a new contact for that customer, then the contact record inherits a primary address from the associated customer record. This happens for vendor contact, too. Dataverse doesn’t currently support this behavior. If dual-write is enabled, a customer contacts that is inherited with a primary address from the Finance and Operations app is synchronized to Dataverse along with its address.

- Electronic addresses set on the electronic address tab of the **Account**, **Contact**, and **Vendor** forms come from the **msdyn_partyelectronicaddresses** table. This information does not flow to its associated transactions like sales order, quotation, and purchase order. We plan to fix this issue in an incremental release. The existing data on the electronic address fields on the account and contact records will continue to work on transactions like sales order, quotation, and purchase order.

- In Finance and Operations apps, you can create a contact record from the **Add Contact** form. When you try to create a new contact from the **View Contact** form, the action fails. This is a known issue.
Initial sync does not support the Available From and Available To time fields on ContactForParty, because DIXF converts the value into a string instead of an integer. The conversion triggers the error "Cannot convert the literal '<say 08:00:00>' to the expected type edm.int32."

When a postal address is used for more than one reason, for example, business communication address and billing address, it should appear as Business;Invoice as shown in the following image. If you add a space in between the values, you will get an error.

You can’t enter a forward-dated postal address using a Finance and Operations app with dual-write, because Dataverse does not support date effectivity. If you enter a future-dated postal address using a Finance and Operations app, it synchronizes to Dataverse fully and you will see the address on the user interface immediately. Any updates to this record will result in an error as it is future-dated and not current in the Finance and Operations app.
The Dual-write application orchestration solution version 2.0.999.0 and later includes data model changes to party and global address book for the Account and Contact tables. The changes allow many-to-many relationships that support advanced business scenarios. These changes are not supported by portal web roles, including the customer portal, that are shipped out-of-the-box or that existed in your environment before you installed dual-write. For the web roles to work as expected, you need to create new web roles by using the new data model.

In summary, the way the tables interact has changed, but the table permissions in the customer portal haven’t changed. This topic explains how to create new web roles that work with the new advanced data model.

This diagram shows the table relationship without the party and global address book data model:

```
Account -> Contacts
```

This diagram shows the table relationship with the party and global address book data model:
Create a new table permission

To create these new table permissions, follow these steps:

1. Sign in to Power Apps, and go to your apps.
2. Select your Portal Management app.
3. In the side bar, select Security > Table permissions.

You must create three new permissions:

- Contact to Party table connection
- Party to Account table connection
- Account to Order table connection

4. Create and save a new permission for the Contact to Party connection, setting these parameters:
   - **Name**: Party to Account table connection (or your choice)
   - **Table Name**: msdyn_contactforparty
5. Create and save a new permission for the Party to Account connection, setting these parameters:

- **Name**: Party to Account Connection (or your choice)
- **Table Name**: account
- **Website**: Customer Portal
- **Scope**: Parent
- **Privileges**: Select all
- **Parent Table Permission**: Contact to Party Connection

6. Create and save a new permission for the Account to Order connection, setting these parameters:

- **Name**: Account to Order Connection (or your choice)
- **Table Name**: salesorder
- **Website**: Customer Portal
- **Scope**: Parent
- **Privileges**: Select all
- **Parent Table Permission**: Party to Account Connection
The Microsoft Azure Data Factory templates help you upgrade the following existing data in dual-write to the party and global address book model: data in the **Account**, **Contact**, and **Vendor** tables, and postal and electronic addresses.

The following three Data Factory templates are provided. They help reconcile the data from both Finance and Operations apps and customer engagement apps.

- **Party template** *(Upgrade data to dual-write Party-GAB schema/arm_template.json)* – This template helps upgrade **Party** and **Contact** data that is associated with **Account**, **Contact**, and **Vendor** data.

- **Party postal address template** *(Upgrade data to dual-write Party-GAB schema/Upgrade to Party Postal Address - GAB/arm_template.json)* – This template helps upgrade the postal addresses that are associated with **Account**, **Contact**, and **Vendor** data.

- **Party electronic address template** *(Upgrade data to dual-write Party-GAB schema/Upgrade to Party Electronic Address - GAB/arm_template.json)* – This template helps upgrade electronic addresses that are associated with **Account**, **Contact**, and **Vendor** data.

At the end of the process, the following comma-separated values (.csv) files are generated.

<table>
<thead>
<tr>
<th>FILE NAME</th>
<th>PURPOSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FONewParty.csv</td>
<td>This file helps create new <strong>Party</strong> records inside the Finance and Operations app.</td>
</tr>
<tr>
<td>ImportFONewPostalAddressLocation.csv</td>
<td>This file helps create new <strong>Postal Address Location</strong> records in the Finance and Operations app.</td>
</tr>
<tr>
<td>ImportFONewPartyPostalAddress.csv</td>
<td>This file helps create new <strong>Party Postal address</strong> records in the Finance and Operations app.</td>
</tr>
<tr>
<td>ImportFONewPostalAddress.csv</td>
<td>This file helps create new <strong>Postal Address</strong> records in the Finance and Operations app.</td>
</tr>
<tr>
<td>ImportFONewElectronicAddress.csv</td>
<td>This file helps create new <strong>Electronic Address</strong> records in the Finance and Operations app.</td>
</tr>
</tbody>
</table>

This topic explains how to use the Data Factory templates and upgrade your data. If you don’t have any customizations, you can use the templates as they are. However, if you have customizations for **Account**, **Contact**, and **Vendor** data, you must modify the templates as described in this topic.

**IMPORTANT**

There are special instructions if you will run the Party postal address and Party electronic address templates. You must run the Party template first, then the Party postal address template, and then the Party electronic address template.
Prerequisites

The following prerequisites must be in place before you can upgrade to the party and global address book model:

- You must have an Azure subscription.
- You must have access to the templates.
- You must be an existing dual-write customer.

Prepare for the upgrade

An upgrade requires the following preparation:

- **Full synchronization**: Both the Finance and operations environment and the customer engagement environment are in a fully synced state for the Account (Customer), Contact, and Vendor tables.
- **Integration keys**: The Account (Customer), Contact, and Vendor tables in customer engagement apps are using the out-of-box integration keys. If you customized the integration keys, you must customize the template.
- **Party number**: All Account (Customer), Contact, and Vendor records that will be upgraded have a party number. Records that don't have a party number will be ignored. If you want to upgrade those records, add a party number to them before you start the upgrade process.
- **System outage**: During the upgrade process, you will have to take both the Finance and operations environment and the customer engagement environment offline.
- **Snapshot**: Take a snapshot of both the Finance and Operations apps and the customer engagement apps. You can then use the snapshots to restore the previous state if you must.

Deployment

1. Download the templates from Dynamics-365-FastTrack-Implementation-Assets.
2. Sign in to the Azure portal.
3. Create a resource group.
4. Create a storage account in the resource group that you created.
5. Create a data factory in the resource group that you created.
6. Open the data factory, and select the Author & Monitor tile.
7. On the Manage tab, select ARM template.
8. Select Import ARM template to import the Party template.
9. Import the template into the data factory. Enter the following values for Project details and Instance details.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscription</td>
<td>The Azure subscription</td>
</tr>
<tr>
<td>Resource group</td>
<td>Provide the same resource that the storage account is created under</td>
</tr>
<tr>
<td>Region</td>
<td>The region</td>
</tr>
<tr>
<td>FIELD</td>
<td>VALUE</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Factory Name</td>
<td>The factory name</td>
</tr>
<tr>
<td>FO Linked Service_service Principal Key</td>
<td>The application's key</td>
</tr>
<tr>
<td>Azure Blob Storage_connection String</td>
<td>The Azure Blob storage connection string</td>
</tr>
<tr>
<td>Dynamics Crm Linked Service_password</td>
<td>The password for the user account that you specify as the user name</td>
</tr>
<tr>
<td>FO Linked Service_properties_type Properties_url</td>
<td><a href="https://sampledynamics.sandbox-operationsdynamics.com/data">https://sampledynamics.sandbox-operationsdynamics.com/data</a></td>
</tr>
<tr>
<td>FO Linked Service_properties_type Properties_tenant</td>
<td>Information (domain name or tenant ID) about the tenant that your application resides under</td>
</tr>
<tr>
<td>FO Linked Service_properties_type Properties_aad Resource Id</td>
<td><a href="https://sampledynamics.sandboxoperationsdynamics.com">https://sampledynamics.sandboxoperationsdynamics.com</a></td>
</tr>
<tr>
<td>FO Linked Service_properties_type Properties_service Principal Id</td>
<td>The application's client ID</td>
</tr>
<tr>
<td>Dynamics Crm Linked Service_properties_type Properties_username</td>
<td>The user name that is used to connect to Dynamics 365</td>
</tr>
</tbody>
</table>

For more information, see the following topics:

- [Manually promote a Resource Manager template for each environment](#)
- [Linked service properties](#)
- [Copy data using Azure Data Factory](#)

10. After deployment, validate the datasets, data flow, and linked service of the data factory.

11. Go to Manage. Under Connections, select Linked Service. Then select DynamicsCrmLinkedService. In the Edit linked service (Dynamics CRM) dialog box, enter the following values.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>DynamicsCrmLinkedService</td>
</tr>
</tbody>
</table>
Prepare to run the Data Factory templates

This section describes the setup that is required before you run the Party postal address and Party electronic address Data Factory templates.

Setup to run the Party postal address template

1. Sign in to customer engagement apps, and go to Settings > Personalization Settings. Then, on the General tab, configure time zone setting for the system admin account. The time zone must be in Coordinated Universal Time (UTC) to update the “valid from” and “valid to” dates of postal addresses from Finance and Operations apps.

   **Set the time zone you are in**

   ![Time Zone Selector](https://example.com/timezone-selector.png)

2. In Data Factory, on the Manage tab, under Global parameters, create the following global parameter.

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>NAME</th>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PostalAddressIdPrefix</td>
<td>string</td>
<td>This parameter appends a serial number to newly created postal addresses as a prefix. Be sure to provide a string that doesn't conflict with postal addresses in Finance and Operations apps and customer engagement apps. For example, use <strong>ADF-PAD-</strong>.</td>
</tr>
</tbody>
</table>
3. When you've finished, select **Publish all**.

### Setup to run the Party electronic address template

1. In Data Factory, on the **Manage** tab, under **Global parameters**, create the following global parameters.

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>NAME</th>
<th>TYPE</th>
<th>VALUE</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IsFOSource</td>
<td>bool</td>
<td></td>
<td>This parameter determines which primary system addresses are replaced in the event of conflicts. If the value is <strong>true</strong>, the primary addresses in Finance and Operations apps will replace the primary addresses in customer engagement apps. If the value is <strong>false</strong>, the primary addresses in customer engagement apps will replace the primary addresses in Finance and Operations apps.</td>
</tr>
<tr>
<td>2</td>
<td>ElectronicAddressIdPrefix</td>
<td>string</td>
<td></td>
<td>This parameter appends a serial number to newly created electronic addresses as a prefix. Be sure to provide a string that doesn't conflict with electronic addresses in Finance and Operations apps and customer engagement apps. For example, use <strong>ADF-EAD-</strong>.</td>
</tr>
</tbody>
</table>

2. When you've finished, select **Publish all**.

### Run the templates

1. Stop the following **Account**, **Contact**, and **Vendor** dual-write maps that use the Finance and Operations
app:

- Customers V3(accounts)
- Customers V3(contacts)
- CDS Contacts V2(contacts)
- CDS Contacts V2(contacts)
- Vendor V2 (msdyn_vendor)

2. Make sure that the maps are removed from the **msdy_dualwriteruntimeconfig** table in Dataverse.

3. Install **Dual-write Party and Global Address Book Solutions** from **AppSource**.

4. In the Finance and Operations app, run **Initial Sync** for the following tables if they contain data:

   - Salutations
   - Personal character types
   - Complimentary closing
   - Contact person titles
   - Decision making roles
   - Loyalty levels

5. In the customer engagement app, disable the following plug-in steps:

   - **Account Update**
     - Microsoft.Dynamics.FinanceExtended.Plugins.TriggerNotesForCustomerTypeCodes: Update of account
   - **Contact Update**
     - Microsoft.Dynamics.GABExtended.Plugins.UpdatePartyAttributesFromContactEntity: Update of contact
     - Microsoft.Dynamics.FinanceExtended.Plugins.TriggerNotesForSellableContact: Update of contact
   - **msdyn_party Update**
   - **msdyn_vendor Update**
   - **Customeraddress**
     - Create
       - Microsoft.Dynamics.GABExtended.Plugins.CreatePartyAddress: Create of customeraddress
     - Update
       - Microsoft.Dynamics.GABExtended.Plugins.CreatePartyAddress: Update of customeraddress
     - Delete
       - Microsoft.Dynamics.GABExtended.Plugins.DeleteCustomerAddress: Delete of customeraddress
- msdyn_partypostaladdress
  - Create
    - Microsoft.Dynamics.GABExtended.Plugins.CreateCustomerAddress: Create of msdyn_partypostaladdress
    - Microsoft.Dynamics.GABExtended.Plugins.PartyPostalAddress: Create of msdyn_partypostaladdress
  - Update
    - Microsoft.Dynamics.GABExtended.Plugins.CreateCustomerAddress: Update of msdyn_partypostaladdress
- msdyn_postaladdress
  - Create
    - Microsoft.Dynamics.GABExtended.Plugins.PostalAddress: Create of msdyn_postaladdress
    - Microsoft.Dynamics.GABExtended.Plugins.PostalAddressPostCreate: Create of msdyn_postaladdress
    - Microsoft.Dynamics.GABExtended.Plugins.UpdateCustomerAddress: Create of msdyn_postaladdress
  - Update
    - Microsoft.Dynamics.GABExtended.Plugins.PostalAddressUpdate: Update of msdyn_postaladdress
    - Microsoft.Dynamics.GABExtended.Plugins.UpdateCustomerAddress: Update of msdyn_postaladdress
- msdyn_partyelectronicaddress
  - Create
    - Microsoft.Dynamics.GABExtended.Plugins.PartyElectronicAddressSync: Create of msdyn_partyelectronicaddress
  - Update
  - Delete

6. In the customer engagement app, disable the following workflows:
   - Create Vendors in Accounts Table
   - Create Vendors in Accounts Table
   - Create Vendors of type person in Contacts Table
   - Create Vendors of type Person in Vendors Table
   - Update Vendors in Accounts Table
   - Update Vendors in Vendors Table
   - Update Vendors of type Person in Contacts Table
   - Update Vendors of type Person in Vendors Table

7. In the data factory, run the template by selecting Trigger now as shown in the following illustration. This
process might take a few hours to be completed, depending on the data volume.

**NOTE**

If you have customizations for Account, Contact, and Vendor, you must modify the template.

8. Import the new Party records into the Finance and Operations app.

   a. Download the `FONewParty.csv` file from Azure Blob storage. The path is `partybootstrapping/output/FONewParty.csv`.

   b. Convert the `FONewParty.csv` file to an Excel file, and import the Excel file into the Finance and Operations app. Alternatively, if the CSV import works for you, you can import the .csv file directly. This step might take a few hours to be completed, depending on the data volume. For more information, see Data import and export jobs overview.

9. In the data factory, run the Party postal address and Party electronic address templates, one after the other.

   - The Party postal address template upserts all postal address records in the customer engagement app, and associates them with corresponding Account, Contact, and Vendor records. It also generates three .csv files: `ImportFONewPostalAddressLocation.csv`, `ImportFONewPartyPostalAddress.csv`, and `ImportFONewPostalAddress.csv`.

   - The Party electronic address template upserts all electronic addresses in the customer engagement app, and associates them with corresponding Account, Contact, and Vendor records. It also generates one .csv file: `ImportFONewElectronicAddress.csv`. 
10. To update the Finance and Operations app with this data, you must convert the .csv files into an Excel workbook and import it into the Finance and Operations app. Alternatively, if the CSV import works for you, you can import the .csv files directly. This step might take a few hours to be completed, depending on the volume.

11. In the customer engagement app, enable the following plug-in steps:

- Account Update
  - Microsoft.Dynamics.FinanceExtended.Plugins.TriggerNotesForCustomerTypeCodes: Update of account
- Contact Update
  - Microsoft.Dynamics.GABExtended.Plugins.UpdatePartyAttributesFromContactEntity: Update of contact
  - Microsoft.Dynamics.FinanceExtended.Plugins.TriggerNotesForSellableContact: Update of contact
- msdyn_party Update
- msdyn_vendor Update
- msdyn_partypostaladdress
  - Create
Explanation of the Data Factory templates

- Microsoft.Dynamics.GABExtended.Plugins.CreateCustomerAddress: Create of msdyn_partypostaladdress
- Microsoft.Dynamics.GABExtended.Plugins.PartyPostalAddress: Create of msdyn_partypostaladdress

- Update
  - Microsoft.Dynamics.GABExtended.Plugins.CreateCustomerAddress: Update of msdyn_partypostaladdress

- msdyn_postaladdress
  - Create
    - Microsoft.Dynamics.GABExtended.Plugins.PostalAddress: Create of msdyn_postaladdress
    - Microsoft.Dynamics.GABExtended.Plugins.PostalAddressPostCreate: Create of msdyn_postaladdress
    - Microsoft.Dynamics.GABExtended.Plugins.UpdateCustomerAddress: Create of msdyn_postaladdress
  
  - Update
    - Microsoft.Dynamics.GABExtended.Plugins.PostalAddressUpdate: Update of msdyn_postaladdress
    - Microsoft.Dynamics.GABExtended.Plugins.UpdateCustomerAddress: Update of msdyn_postaladdress

- msdyn_partyelectronicaddress
  - Create
    - Microsoft.Dynamics.GABExtended.Plugins.PartyElectronicAddressSync: Create of msdyn_partyelectronicaddress
  
  - Update
  
  - Delete

12. In the customer engagement app, activate the following workflows if you previously inactivated them:

- Create Vendors in Accounts Table
- Create Vendors in Accounts Table
- Create Vendors of type person in Contacts Table
- Create Vendors of type Person in Vendors Table
- Update Vendors in Accounts Table
- Update Vendors in Vendors Table
- Update Vendors of type Person in Contacts Table
- Update Vendors of type Person in Vendors Table

13. Run the **Party** record–related maps as described in *Party and global address book.*
This section takes you through the steps in each Data Factory template.

**Steps in the Party template**

1. Steps 1 through 6 identify the companies that are enabled for dual-write and builds a filter clause for them.

2. Steps 7-1 through 7-9 retrieve data from both the Finance and Operations app and the customer engagement app, and stage that data for upgrade.

3. Steps 8 through 9 compare the party number for **Account**, **Contact**, and **Vendor** records between the Finance and Operations app and the customer engagement app. Any records that don't have a party number are skipped.

4. Step 10 generates two .csv files for the party records that must be created in the customer engagement app and the Finance and Operations app.

   - **FOCDSParty.csv** – This file contains all party records of both systems, regardless of whether the company is enabled for dual-write.
   - **FONewParty.csv** – This file contains a subset of the party records that Dataverse is aware of (for example, accounts of the **Prospect** type).

5. Step 11 creates the parties in the customer engagement app.

6. Step 12 retrieves the globally unique identifiers (GUIDs) of the parties from the customer engagement app and stages them so that they can be associated with **Account**, **Contact**, and **Vendor** records in subsequent steps.

7. Step 13 associates the **Account**, **Contact**, and **Vendor** records with party GUIDs.

8. Steps 14-1 through 14-3 update the **Account**, **Contact**, and **Vendor** records in the customer engagement app with party GUIDs.

9. Steps 15-1 through 15-3 prepare **Contact for Party** records for **Account**, **Contact**, and **Vendor** records.

10. Steps 16-1 through 16-7 retrieve reference data such as salutations and personal character types, and associate it with **Contact for Party** records.

11. Step 17 merges the **Contact for Party** records for **Account**, **Contact**, and **Vendor** records.

12. Step 18 imports the **Contact for Party** records into the customer engagement app.

**Steps in the Party postal address template**

1. Steps 1-1 through 1-10 retrieve data from both the Finance and Operations app and the customer engagement app, and stage that data for upgrade.

2. Step 2 de-normalizes the postal address data in the Finance and Operations app by joining the postal address and the party postal address.

3. Step 3 de-duplicates and merges account, contact, and vendor address data from the customer engagement app.

4. Step 4 creates .csv files for the Finance and Operations app to create new address data that is based on account, contact, and vendor addresses.

5. Step 5-1 creates .csv files for the customer engagement app to create all address data, based on both the Finance and Operations app and the customer engagement app.


   - **ImportFONewPostalAddressLocation.csv**
Steps in the Party electronic address template

1. Steps 1-1 through 1-5 retrieve data from both the Finance and Operations app and the customer engagement app, and stage that data for upgrade.

2. Step 2 consolidates electronic addresses in the customer engagement app from account, contact, and vendor entities.

3. Step 3 merges primary electronic address data from the customer engagement app and the Finance and Operations app.

4. Step 4 creates .csv files.
   - Create new electronic address data for the Finance and Operations app, based on account, contact, and vendor addresses.
   - Create new electronic address data for the customer engagement app, based on electronic address, account, contact and vendor addresses in the Finance and Operations app.

5. Step 5-1 imports electronic addresses into the customer engagement app.

6. Step 5-2 creates .csv files to update primary addresses for accounts and contacts in the customer engagement app.

7. Steps 6-1 through 6-2 import accounts and contact primary addresses into the customer engagement app.

Troubleshooting

1. If the process fails, rerun the data factory. Start from the failed activity.

2. Some files that are generated by the data factory can be used for data validation.

3. The data factory runs based on .csv files. Therefore, if a comma is included in any field value, it might interfere with the results. You must remove all commas from field values.

4. The Monitoring tab provides information about all steps and data that have been processed. Select a specific step to debug it.
Learn more about the template

For more information about the template, see Comments for Azure Data Factory template readme.
During business processes, Microsoft Dynamics 365 users often gather information about their customers. This information is recorded as activities and notes. This topic describes the integration of note data in dual-write.

Customer information can be classified in the following ways:

- **Actionable information that a Dynamics 365 user handles on behalf of a customer** – For example, Contoso (a Dynamics 365 user) is conducting a game show. One of Contoso’s customers (a customer) wants to attend the game show. The customer asks a Contoso employee to book a slot in the game show for them. The booking occurs in Contoso’s event attendee’s calendar.

- **Actionable information for a Dynamics 365 user** – For example, a customer who is purchasing a Surface unit enters special instructions that indicate that the device should be gift wrapped before delivery. These instructions are actionable information that should be handled by the Contoso employee who is responsible for packaging.

- **Non-actionable information** – For example, a customer visits the Contoso store and, during their conversation with a store associate, expresses interest in *Halo* games and gaming accessories. The store associate makes a note of this information. The product recommendations engine then uses it to make recommendations to the customer.

In general, actionable information is captured as *activities* in Finance and Operations apps and customer engagement apps. Non-actionable information is captured as *notes* in Finance and Operations apps, and as *annotations* in customer engagement apps.

**TIP**

Although notes are intended for non-actionable information, the apps won’t prevent you from using them to store and handle actionable information if you want to use them in that way.

Microsoft is currently releasing functionality for note integration. (Functionality for activity integration will be released later.) Note integration is available for customers, vendors, sales orders, and purchase orders.

**Create a note in a customer engagement app**

To create a note in a customer engagement app and then sync it to a Finance and Operations app, follow these steps.

1. In the customer engagement app, open the account record for a customer.

2. In the **Timeline** pane, select the plus sign (+), and then select **Note** to create a note.
3. Enter a title and description, and then select **Add note**.

The new note is added to the customer timeline.

4. Sign in to the Finance and Operations app, and open the same customer record. Notice that the **Attachments** button (paperclip symbol) in the upper-right corner indicates that the record has an attachment.

5. Select the **Attachments** button to open the **Attachments** page. You should find the note that you created in the customer engagement app.
Create a note in a Finance and Operations app

You can also create a note in a Finance and Operations app, and it will be synced to a customer engagement app.

To create a note in a Finance and Operations app and then sync it to a customer engagement app, follow these steps.

1. In the Finance and Operations app, on the **Attachments** page, select **New > Note**.

2. Enter a title and a brief set of instructions, and then select **Save**.

3. In the customer engagement app, update the record. You should find the new note on the timeline.
You can classify a note as either internal or external.

- In the Finance and Operations app, on the Attachments page, open the note, and then, in the Restriction field, select **Internal** or **External**.

You can also create a URL.

1. In the Finance and Operations app, on the Attachments page, select **New > URL**.
2. Enter a title and the URL.
3. In the Restriction field, select **Internal** or **External**.
4. Select **Save**.

Because customer engagement apps don't have a URL type, the URL is integrated with dual-write as a note.
NOTE

File attachments aren't supported.

Templates

Note integration includes a collection of table maps that work together during data interaction, as shown in the following table.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APP</th>
<th>CUSTOMER ENGAGEMENT APP</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer Attachments</td>
<td>Annotations</td>
<td>Businesses that use plain text and URLs to capture customer-specific information (for both organizations and persons).</td>
</tr>
<tr>
<td>Vendor document attachments</td>
<td>Annotations</td>
<td>Businesses that use plain text and URLs to capture vendor-specific information (for both organizations and persons).</td>
</tr>
<tr>
<td>Sales order header document attachments</td>
<td>Annotations</td>
<td>Businesses that use plain text and URLs to capture sales order–specific information.</td>
</tr>
<tr>
<td>Purchase order header document attachments</td>
<td>Annotations</td>
<td>Businesses that use plain text and URLs to capture purchase order–specific information.</td>
</tr>
</tbody>
</table>

Limitations

Once you install the notes solution, you cannot uninstall it.

For more information, see Dual-write mapping reference.
Mapping tables

Mapping types
There are several different mapping types. The following table explains the symbols used in the template tables.

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>One-way</td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>One-way, and data is transformed in the process.</td>
</tr>
<tr>
<td>=</td>
<td>Bidirectional</td>
</tr>
<tr>
<td>&gt; &lt;</td>
<td>Bidirectional, and data is transformed in the process.</td>
</tr>
<tr>
<td>&lt; &lt;</td>
<td>One-way, and data is transformed in the process.</td>
</tr>
</tbody>
</table>

Filters
The source filter and reverse source filter determine which rows are synchronized.

Default values
If a synchronized field does not exist in either the finance and operations table or the customer engagement table, then a default value is assigned in the synchronized table. In some cases, the default value is an integer that is a lookup to an attribute value in Dataverse. For example, in the Contact table, the default value for address1AddressTypeCode is 3 (Primary address).

Templates

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APP</th>
<th>CUSTOMER ENGAGEMENT APP</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>All products</td>
<td>msdyn_globalproducts</td>
<td></td>
</tr>
<tr>
<td>Asset management asset lifecycle models</td>
<td>msdyn_assetlifecyclemodels</td>
<td></td>
</tr>
<tr>
<td>Asset management asset lifecycle states</td>
<td>msdyn_assetlifecyclestates</td>
<td></td>
</tr>
<tr>
<td>Asset management asset types</td>
<td>msdyn_customerassetcategories</td>
<td></td>
</tr>
<tr>
<td>Asset management assets</td>
<td>msdyn_customerassets</td>
<td></td>
</tr>
<tr>
<td>Asset management functional location lifecycle models</td>
<td>msdyn_functionallocationlifecyclemodels</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS APP</td>
<td>CUSTOMER ENGAGEMENT APP</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------------------------</td>
<td>--------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Asset management functional location lifecycle states</td>
<td>msdyn_functionallocationlifecyclestates</td>
<td></td>
</tr>
<tr>
<td>Asset management functional location types</td>
<td>msdyn_functionallocationtypes</td>
<td></td>
</tr>
<tr>
<td>Asset management functional locations</td>
<td>msdyn_functionallocations</td>
<td></td>
</tr>
<tr>
<td>Asset management manufacturers</td>
<td>msdyn_manufacturers</td>
<td></td>
</tr>
<tr>
<td>Asset management models</td>
<td>msdyn_models</td>
<td></td>
</tr>
<tr>
<td>Asset management warranty</td>
<td>msdyn_warranties</td>
<td></td>
</tr>
<tr>
<td>CDS Contacts V2</td>
<td>contacts</td>
<td></td>
</tr>
<tr>
<td>CDS Exchange Rates</td>
<td>msdyn_currencyexchangerates</td>
<td></td>
</tr>
<tr>
<td>CDS Parties</td>
<td>msdyn_parties</td>
<td></td>
</tr>
<tr>
<td>CDS Party postal address locations</td>
<td>msdyn_party postal addresses</td>
<td></td>
</tr>
<tr>
<td>CDS inventory on-hand entries</td>
<td>msdyn_inventoryonhandentries</td>
<td></td>
</tr>
<tr>
<td>CDS inventory on-hand requests</td>
<td>msdyn_inventoryonhandrequests</td>
<td></td>
</tr>
<tr>
<td>CDS postal address history V2</td>
<td>msdyn_postaladdresses</td>
<td></td>
</tr>
<tr>
<td>CDS postal address locations</td>
<td>msdyn_postaladdresscollections</td>
<td></td>
</tr>
<tr>
<td>CDS purchase order line entity</td>
<td>msdyn_purchaseorderproducts</td>
<td></td>
</tr>
<tr>
<td>CDS purchase order line soft deleted entity</td>
<td>msdyn_purchaseorderproducts</td>
<td></td>
</tr>
<tr>
<td>CDS released distinct products</td>
<td>products</td>
<td></td>
</tr>
<tr>
<td>CDS sales order headers</td>
<td>salesorders</td>
<td></td>
</tr>
<tr>
<td>CDS sales order lines</td>
<td>salesorderdetails</td>
<td></td>
</tr>
<tr>
<td>CDS sales quotation header</td>
<td>quotes</td>
<td></td>
</tr>
<tr>
<td>CDS sales quotation lines</td>
<td>quotedetails</td>
<td></td>
</tr>
<tr>
<td>Chart of accounts</td>
<td>msdyn_chartofaccountses</td>
<td></td>
</tr>
<tr>
<td>Colors</td>
<td>msdyn_productcolors</td>
<td></td>
</tr>
<tr>
<td>Compensation job function</td>
<td>cdm_jobfunctions</td>
<td></td>
</tr>
<tr>
<td>Finance and Operations App</td>
<td>Customer Engagement App</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Compensation job type</td>
<td>cdm_jobtypes</td>
<td></td>
</tr>
<tr>
<td>Complimentary closings</td>
<td>msdyn_complimentaryclosings</td>
<td></td>
</tr>
<tr>
<td>Configurations</td>
<td>msdyn_productconfigurations</td>
<td></td>
</tr>
<tr>
<td>Contact person titles</td>
<td>msdyn_salescontactpersontitles</td>
<td></td>
</tr>
<tr>
<td>Contacts V2</td>
<td>msdyn_contactforparties</td>
<td></td>
</tr>
<tr>
<td>Currencies</td>
<td>transactioncurrencies</td>
<td></td>
</tr>
<tr>
<td>Customer Attachments</td>
<td>annotations</td>
<td></td>
</tr>
<tr>
<td>Customer groups</td>
<td>msdyn_customergroups</td>
<td></td>
</tr>
<tr>
<td>Customer hierarchies</td>
<td>msdyn_customerhierarchies</td>
<td></td>
</tr>
<tr>
<td>Customer hierarchy nodes</td>
<td>msdyn_customerhierarchynodes</td>
<td></td>
</tr>
<tr>
<td>Customer payment method</td>
<td>msdyn_customerpaymentmethodsoptions</td>
<td></td>
</tr>
<tr>
<td>Customers V3</td>
<td>accounts</td>
<td></td>
</tr>
<tr>
<td>Employment job functions</td>
<td>msdyn_employmentjobfunctions</td>
<td></td>
</tr>
<tr>
<td>Employment per company</td>
<td>cdm_employments</td>
<td></td>
</tr>
<tr>
<td>Ethnic origins</td>
<td>cdm_ethnicorigins</td>
<td></td>
</tr>
<tr>
<td>Exchange rate currency pair</td>
<td>msdyn_currencyexchangeratepairs</td>
<td></td>
</tr>
<tr>
<td>Exchange rate type</td>
<td>msdyn_exchangeratetypes</td>
<td></td>
</tr>
<tr>
<td>Financial dimension format</td>
<td>msdyn_financialdimensionformats</td>
<td></td>
</tr>
<tr>
<td>Financial dimensions</td>
<td>msdyn_dimensionattributes</td>
<td></td>
</tr>
<tr>
<td>Fiscal calendar integration entity</td>
<td>msdyn_fiscalcalendars</td>
<td></td>
</tr>
<tr>
<td>Fiscal calendar period</td>
<td>msdyn_fiscalcalendarperiods</td>
<td></td>
</tr>
<tr>
<td>Fiscal calendar year integration entity</td>
<td>msdyn_fiscalcalendaryears</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS APP</td>
<td>CUSTOMER ENGAGEMENT APP</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>---------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Inventory aisle</td>
<td>msdyn_warehouseaisles</td>
<td></td>
</tr>
<tr>
<td>Item sales tax group</td>
<td>msdyn_taxitemgroups</td>
<td></td>
</tr>
<tr>
<td>Jobs</td>
<td>cdm_jobs</td>
<td></td>
</tr>
<tr>
<td>Language codes</td>
<td>cdm_languages</td>
<td></td>
</tr>
<tr>
<td>Ledger</td>
<td>msdyn_ledgers</td>
<td></td>
</tr>
<tr>
<td>Legal entities</td>
<td>cdm_companies</td>
<td></td>
</tr>
<tr>
<td>Legal entities</td>
<td>msdyn_internalorganizations</td>
<td></td>
</tr>
<tr>
<td>Loyalty card</td>
<td>msdyn_loyaltycards</td>
<td></td>
</tr>
<tr>
<td>Loyalty levels</td>
<td>msdyn_loyaltylevels</td>
<td></td>
</tr>
<tr>
<td>Loyalty reward points</td>
<td>msdyn_loyaltyrewardpoints</td>
<td></td>
</tr>
<tr>
<td>Main account</td>
<td>msdyn_mainaccounts</td>
<td></td>
</tr>
<tr>
<td>Main account categories</td>
<td>msdyn_mainaccountcategories</td>
<td></td>
</tr>
<tr>
<td>Mixed reality guides entity</td>
<td>msmrw_guides</td>
<td></td>
</tr>
<tr>
<td>Modes of delivery</td>
<td>msdyn_shipvias</td>
<td></td>
</tr>
<tr>
<td>Name affixes</td>
<td>msdyn_nameaffixes</td>
<td></td>
</tr>
<tr>
<td>Operating unit</td>
<td>msdyn_internalorganizations</td>
<td></td>
</tr>
<tr>
<td>Organization hierarchy - published</td>
<td>msdyn_internalorganizationhierarchies</td>
<td></td>
</tr>
<tr>
<td>Organization hierarchy purposes</td>
<td>msdyn_internalorganizationhierarchypurposes</td>
<td></td>
</tr>
<tr>
<td>Organization hierarchy type</td>
<td>msdyn_internalorganizationhierarchytypes</td>
<td></td>
</tr>
<tr>
<td>Party contacts V3</td>
<td>msdyn_partyelectronicaddresses</td>
<td></td>
</tr>
<tr>
<td>Payment day lines CDS V2</td>
<td>msdyn_paymentdaylines</td>
<td></td>
</tr>
<tr>
<td>Payment days CDS</td>
<td>msdyn_paymentdays</td>
<td></td>
</tr>
<tr>
<td>Payment schedule</td>
<td>msdyn_paymentschedules</td>
<td></td>
</tr>
<tr>
<td>Payment schedule lines</td>
<td>msdyn_paymentschedulelines</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS APP</td>
<td>CUSTOMER ENGAGEMENT APP</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>---------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Personal character types</td>
<td>msdyn_personalcharactertypes</td>
<td></td>
</tr>
<tr>
<td>Position type</td>
<td>cdm_positiontypes</td>
<td></td>
</tr>
<tr>
<td>Position worker assignments</td>
<td>cdm_positionworkerassignmentmaps</td>
<td></td>
</tr>
<tr>
<td>Positions V2</td>
<td>cdm_jobpositions</td>
<td></td>
</tr>
<tr>
<td>Price customer groups</td>
<td>msdyn_pricecustomergroups</td>
<td></td>
</tr>
<tr>
<td>Product Number Identified Barcode</td>
<td>msdyn_productbarcodes</td>
<td></td>
</tr>
<tr>
<td>Product categories</td>
<td>msdyn_productcategories</td>
<td></td>
</tr>
<tr>
<td>Product category assignments</td>
<td>msdyn_productcategoryassignments</td>
<td></td>
</tr>
<tr>
<td>Product category hierarchies</td>
<td>msdyn_productcategoryhierarchies</td>
<td></td>
</tr>
<tr>
<td>Product category hierarchy roles</td>
<td>msdyn_productcategoryhierarchyroles</td>
<td></td>
</tr>
<tr>
<td>Product default order settings V2</td>
<td>msdyn_productspecificdefaultordersettings</td>
<td></td>
</tr>
<tr>
<td>Product dimension groups</td>
<td>msdyn_productdimensiongroups</td>
<td></td>
</tr>
<tr>
<td>Product master colors</td>
<td>msdyn_sharedproductcolors</td>
<td></td>
</tr>
<tr>
<td>Product master configurations</td>
<td>msdyn_sharedproductconfigurations</td>
<td></td>
</tr>
<tr>
<td>Product master sizes</td>
<td>msdyn_sharedproductsizes</td>
<td></td>
</tr>
<tr>
<td>Product master styles</td>
<td>msdyn_sharedproductstyles</td>
<td></td>
</tr>
<tr>
<td>Product receipt header</td>
<td>msdyn_purchaseorderreceipts</td>
<td></td>
</tr>
<tr>
<td>Product receipt line</td>
<td>msdyn_purchaseorderreceiptproducts</td>
<td></td>
</tr>
<tr>
<td>Product specific unit conversions</td>
<td>msdyn_productspecificunitofmeasureconversions</td>
<td></td>
</tr>
<tr>
<td>Prospects</td>
<td>leads</td>
<td></td>
</tr>
<tr>
<td>Purchase order header document attachments</td>
<td>annotations</td>
<td></td>
</tr>
<tr>
<td>Purchase order headers V2</td>
<td>msdyn_purchaseorders</td>
<td></td>
</tr>
<tr>
<td>Released products V2</td>
<td>msdyn_sharedproductdetails</td>
<td></td>
</tr>
<tr>
<td>Sales invoice headers V2</td>
<td>invoices</td>
<td></td>
</tr>
<tr>
<td><strong>FINANCE AND OPERATIONS APP</strong></td>
<td><strong>CUSTOMER ENGAGEMENT APP</strong></td>
<td><strong>DESCRIPTION</strong></td>
</tr>
<tr>
<td>--------------------------------------------------------</td>
<td>---------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Sales invoice lines V2</td>
<td>invoicedetails</td>
<td></td>
</tr>
<tr>
<td>Sales order header document attachments</td>
<td>annotations</td>
<td></td>
</tr>
<tr>
<td>Sales order origin codes</td>
<td>msdyn_salesorderorigins</td>
<td></td>
</tr>
<tr>
<td>Sales tax authorities</td>
<td>msdyn_taxauthorities</td>
<td></td>
</tr>
<tr>
<td>Sales tax exempt code entity CDS</td>
<td>msdyn_taxexemptcodes</td>
<td></td>
</tr>
<tr>
<td>Sales tax groups</td>
<td>msdyn_taxgroups</td>
<td></td>
</tr>
<tr>
<td>Sales tax ledger posting groups V2</td>
<td>msdyn_taxpostinggroups</td>
<td></td>
</tr>
<tr>
<td>Salutations</td>
<td>msdyn_salutations</td>
<td></td>
</tr>
<tr>
<td>Sites</td>
<td>msdyn_operationalssites</td>
<td></td>
</tr>
<tr>
<td>Sizes</td>
<td>msdyn_productsizes</td>
<td></td>
</tr>
<tr>
<td>Storage dimension groups</td>
<td>msdyn_productstoragedimensiongroups</td>
<td></td>
</tr>
<tr>
<td>Styles</td>
<td>msdyn_productstyles</td>
<td></td>
</tr>
<tr>
<td>Terms of delivery</td>
<td>msdyn_termsofdeliveries</td>
<td></td>
</tr>
<tr>
<td>Terms of payment</td>
<td>msdyn_paymentterms</td>
<td></td>
</tr>
<tr>
<td>Tracking dimension groups</td>
<td>msdyn_producttrackingdimensiongroups</td>
<td></td>
</tr>
<tr>
<td>Unit conversions</td>
<td>msdyn_unitofmeasureconversions</td>
<td></td>
</tr>
<tr>
<td>Units</td>
<td>uoms</td>
<td></td>
</tr>
<tr>
<td>Vendor document attachments</td>
<td>annotations</td>
<td></td>
</tr>
<tr>
<td>Vendor groups</td>
<td>msdyn_vendorgroups</td>
<td></td>
</tr>
<tr>
<td>Vendor payment method</td>
<td>msdyn_vendorpaymentmethods</td>
<td></td>
</tr>
<tr>
<td>Vendors V2</td>
<td>msdyn_vendors</td>
<td></td>
</tr>
<tr>
<td>Veteran status</td>
<td>cdm_veteranstatuses</td>
<td></td>
</tr>
<tr>
<td>Warehouse locations</td>
<td>msdyn_inventorylocations</td>
<td></td>
</tr>
<tr>
<td>Warehouse work headers</td>
<td>msdyn_warehouseworkheaders</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS APP</td>
<td>CUSTOMER ENGAGEMENT APP</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Warehouse work lines</td>
<td>msdyn_warehouseworklines</td>
<td></td>
</tr>
<tr>
<td>Warehouse zone groups</td>
<td>msdyn_warehousezonegroups</td>
<td></td>
</tr>
<tr>
<td>Warehouse zones</td>
<td>msdyn_warehousezones</td>
<td></td>
</tr>
<tr>
<td>Warehouses</td>
<td>msdyn_warehouses</td>
<td></td>
</tr>
<tr>
<td>Withholding tax codes</td>
<td>msdyn_withholdingtaxcodes</td>
<td></td>
</tr>
<tr>
<td>Withholding tax groups</td>
<td>msdyn_withholdingtaxgroups</td>
<td></td>
</tr>
<tr>
<td>Worker</td>
<td>cdm_workers</td>
<td></td>
</tr>
</tbody>
</table>

**All products (msdyn_globalproducts)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCTNAME</td>
<td>&gt;</td>
<td>msdyn_productname</td>
<td></td>
</tr>
<tr>
<td>PRODUCTNUMBER</td>
<td>&gt;</td>
<td>msdyn_productnumber</td>
<td></td>
</tr>
</tbody>
</table>

**Asset management asset lifecycle models (msdyn_assetlifecyclemodels)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVELIFECYCLESTATEID</td>
<td>=</td>
<td>msdyn_activeifecyclestate. msdyn_assetlifecyclestate_id</td>
<td></td>
</tr>
<tr>
<td>INBOUNDLIFECYCLESTATEID</td>
<td>=</td>
<td>msdyn_inboundifecyclestate.msdyn_assetlifecyclestate_id</td>
<td></td>
</tr>
<tr>
<td>INSTORAGELIFECYCLESTATEID</td>
<td>=</td>
<td>msdyn_instoragelifecyclestate.msdyn_assetlifecyclestate_id</td>
<td></td>
</tr>
<tr>
<td>LIFECYCLEMODELID</td>
<td>=</td>
<td>msdyn_assetlifecyclemodel_id</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>ONLOANLIFECYCLESTATEID</td>
<td>=</td>
<td>msdyn_onloanifecyclestate.msdyn_assetlifecyclestate_id</td>
<td></td>
</tr>
<tr>
<td>OUTBOUNDLIFECYCLESTATEID</td>
<td>=</td>
<td>msdyn_outboundifecyclestate.msdyn_assetlifecyclestate_id</td>
<td></td>
</tr>
<tr>
<td>Finance and Operations Field</td>
<td>Map Type</td>
<td>Customer Engagement Column</td>
<td>Default Value</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Received Lifecycle State ID</td>
<td>=</td>
<td>msdyn_receivedlifecyclestate.msdyn_assetlifecyclestate_id</td>
<td></td>
</tr>
</tbody>
</table>

**Asset management asset lifecycle states (msdyn_assetlifecyclestates)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete Open Calendar Lines</td>
<td>&gt; &lt;</td>
<td>msdyn_deleteopencalendarlines</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>Maintenance Asset Active</td>
<td>&gt; &lt;</td>
<td>msdyn_maintenanceassetactive</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
</tbody>
</table>

**Asset management asset types (msdyn_customerassetcategories)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lifecycle Model ID</td>
<td>=</td>
<td>msdyn_lifecyclemodel.msdyn_assetlifecyclemodel_id</td>
<td></td>
</tr>
<tr>
<td>Maintenance Asset Type ID</td>
<td>=</td>
<td>msdyn_maintenanceassettypeid</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>Calculate KPITotal</td>
<td>&gt; &lt;</td>
<td>msdyn_calculatekpitotal</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td>True</td>
<td></td>
</tr>
</tbody>
</table>

**Asset management assets (msdyn_customerassets)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acquisition Cost</td>
<td>=</td>
<td>msdyn_acquisitioncost</td>
<td></td>
</tr>
<tr>
<td>Acquisition Date</td>
<td>=</td>
<td>msdyn_acquisitiondate</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Map Type</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------</td>
<td>-----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>ACTIVENAME</td>
<td>=</td>
<td>msdyn_activefrom</td>
<td></td>
</tr>
<tr>
<td>ACTIVETO</td>
<td>=</td>
<td>msdyn_activeto</td>
<td></td>
</tr>
<tr>
<td>FUNCTIONALLOCATIONID</td>
<td>=</td>
<td>msdyn_functionallocation_id</td>
<td></td>
</tr>
<tr>
<td>MAINTENANCEASSETLIFECYCLESTATEID</td>
<td>=</td>
<td>msdyn_assetlifecyclestate_id</td>
<td></td>
</tr>
<tr>
<td>MODELID</td>
<td>=</td>
<td>msdyn_model.msdyn_model_id</td>
<td></td>
</tr>
<tr>
<td>MODELPRODUCTID</td>
<td>=</td>
<td>msdyn_model.msdyn_manufacturer.msdyn_manufacturer_id</td>
<td></td>
</tr>
<tr>
<td>MODELYEAR</td>
<td>=</td>
<td>msdyn_modelyear</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>NOTES</td>
<td>=</td>
<td>msdyn_notes</td>
<td></td>
</tr>
<tr>
<td>PURCHASEORDERID</td>
<td>=</td>
<td>msdyn_purchaseorderid</td>
<td></td>
</tr>
<tr>
<td>REPLACEMENTDATE</td>
<td>&gt; &lt; 1/1/1900:blank</td>
<td>msdyn_replacementdate</td>
<td></td>
</tr>
<tr>
<td>REPLACEMENTVALUE</td>
<td>=</td>
<td>msdyn_replacementvalue</td>
<td></td>
</tr>
<tr>
<td>SERIALID</td>
<td>=</td>
<td>msdyn_serialid</td>
<td></td>
</tr>
<tr>
<td>VENDACCOUNT</td>
<td>=</td>
<td>msdyn_vendaccount</td>
<td></td>
</tr>
<tr>
<td>WARRANTYDATEFROMVENDOR</td>
<td>=</td>
<td>msdyn_warrantydatefromvendor</td>
<td></td>
</tr>
<tr>
<td>WARRANTYID</td>
<td>=</td>
<td>msdyn_warranty.msdyn_warranty_id</td>
<td></td>
</tr>
<tr>
<td>WRKCTRID</td>
<td>=</td>
<td>msdyn_wrkctrid</td>
<td></td>
</tr>
<tr>
<td>FIXEDASSETID</td>
<td>=</td>
<td>msdyn_fixedassetid</td>
<td></td>
</tr>
<tr>
<td>MAINTENANCEASSETID</td>
<td>=</td>
<td>msdyn_maintenanceassetid</td>
<td></td>
</tr>
<tr>
<td>PARENTMAINTENANCEASSETID</td>
<td>=</td>
<td>msdyn_parentasset.msdyn_maintenanceassetid</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>-----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>MAINTENANCEASSETTYPEID</td>
<td>=</td>
<td>msdyn_customerassetcategory.msdyn_maintenanceassettypeid</td>
<td></td>
</tr>
<tr>
<td>PRODUCTID</td>
<td>=</td>
<td>msdyn_manufacturer.msdyn_manufacturer_id</td>
<td></td>
</tr>
</tbody>
</table>

**Asset management functional location lifecycle models (msdyn_functionallocationlifecyclemodels)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIFECYCLEMODELID</td>
<td>=</td>
<td>msdyn_functionallocationlifecyclemodel_id</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
</tbody>
</table>

**Asset management functional location lifecycle states (msdyn_functionallocationlifecyclestates)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLOWDELETELOCATION</td>
<td>&gt; &lt;</td>
<td>msdyn_allowdeletelocation</td>
<td></td>
</tr>
<tr>
<td>yes</td>
<td>True</td>
<td>msdyn_allowdeletelocation</td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>False</td>
<td>msdyn_allowdeletelocation</td>
<td></td>
</tr>
<tr>
<td>ALLOWNEWSUBLOCATION</td>
<td>&gt; &lt;</td>
<td>msdyn_allownewsublocations</td>
<td></td>
</tr>
<tr>
<td>yes</td>
<td>True</td>
<td>msdyn_allownewsublocations</td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>False</td>
<td>msdyn_allownewsublocations</td>
<td></td>
</tr>
<tr>
<td>ALLOWRENAMELOCATION</td>
<td>&gt; &lt;</td>
<td>msdyn_allowrenameloaction</td>
<td></td>
</tr>
<tr>
<td>yes</td>
<td>True</td>
<td>msdyn_allowrenameloaction</td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>False</td>
<td>msdyn_allowrenameloaction</td>
<td></td>
</tr>
<tr>
<td>FUNCTIONALLOCATIONACTIVE</td>
<td>&gt; &lt;</td>
<td>msdyn_functionallocationactive</td>
<td></td>
</tr>
<tr>
<td>yes</td>
<td>True</td>
<td>msdyn_functionallocationactive</td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>False</td>
<td>msdyn_functionallocationactive</td>
<td></td>
</tr>
<tr>
<td>LIFECYCLESTATEID</td>
<td>=</td>
<td>msdyn_functionallocationlifecyclestate_id</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>ALLOWINSTALLMAINTENANCEASSETS</td>
<td>&gt; &lt;</td>
<td>msdyn_allowinstallmaintenanceassets</td>
<td></td>
</tr>
<tr>
<td>yes</td>
<td>True</td>
<td>msdyn_allowinstallmaintenanceassets</td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>False</td>
<td>msdyn_allowinstallmaintenanceassets</td>
<td></td>
</tr>
<tr>
<td>CREATELOCATIONMAINTENANCEASSET</td>
<td>&gt; &lt;</td>
<td>msdyn_createlocationmaintenanceasset</td>
<td></td>
</tr>
<tr>
<td>yes</td>
<td>True</td>
<td>msdyn_createlocationmaintenanceasset</td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>False</td>
<td>msdyn_createlocationmaintenanceasset</td>
<td></td>
</tr>
</tbody>
</table>

This template synchronizes data between Finance and Operations apps and Dataverse.
<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAINTENANCEASSETLIFECYCLESTATEID</td>
<td>=</td>
<td>msdyn_assetlifecyclestate.msdyn_assetlifecyclestate_id</td>
<td></td>
</tr>
</tbody>
</table>

**Asset management functional location types (msdyn_functionallocationtypes)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLOWMULTIPLEINSTALLEDASSETS</td>
<td>&gt; &lt;</td>
<td>msdyn_allowmultipleinstalledassets</td>
<td>msdyn_allowmultipleinstalledassets</td>
</tr>
<tr>
<td>FUNCTIONALLOCATIONSID</td>
<td>=</td>
<td>msdyn_functionallocationtype_id</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>UPDATEASSETDIMENSION</td>
<td>&gt; &lt;</td>
<td>msdyn_updateassetdimension</td>
<td>msdyn_updateassetdimension</td>
</tr>
<tr>
<td>LIFECYCLEMODELID</td>
<td>=</td>
<td>msdyn_lifecyclemodelid.msdyn_functionallocationlifecyclemodel_id</td>
<td></td>
</tr>
<tr>
<td>MAINTENANCEASSETTYPEID</td>
<td>=</td>
<td>msdyn_maintenanceassettypeid.msdyn_maintenanceassettypeid</td>
<td></td>
</tr>
</tbody>
</table>

**Asset management functional locations (msdyn_functionallocations)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNCTIONALLOCATIONID</td>
<td>=</td>
<td>msdyn_functionallocation_id</td>
<td></td>
</tr>
<tr>
<td>INVENTORYLOCATIONID</td>
<td>=</td>
<td>msdyn_inventorylocationid</td>
<td></td>
</tr>
<tr>
<td>INVENTORYSITEID</td>
<td>=</td>
<td>msdyn_inventorysiteid</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>NOTES</td>
<td>=</td>
<td>msdyn_notes</td>
<td></td>
</tr>
<tr>
<td>PARENTFUNCTIONALLOCATIONID</td>
<td>=</td>
<td>msdyn_parentfunctionallocation.msdyn_functionallocation_id</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>FUNCTIONALLOCATIONLIFECYCLESTATEID</td>
<td>=</td>
<td>msdyn_functionallocationlifecyclestate.msdyn_functionallocationlifecyclestate_id</td>
<td></td>
</tr>
<tr>
<td>FUNCTIONALLOCATIONTYPEID</td>
<td>=</td>
<td>msdyn_functionallocationtype.msdyn_functionallocationtype_id</td>
<td></td>
</tr>
</tbody>
</table>

**Asset management manufacturers (msdyn_manufacturers)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
<tr>
<td>PRODUCTID</td>
<td>=</td>
<td>msdyn_manufacturer_id</td>
<td></td>
</tr>
</tbody>
</table>

**Asset management models (msdyn_models)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
<tr>
<td>MODELID</td>
<td>=</td>
<td>msdyn_model_id</td>
<td></td>
</tr>
<tr>
<td>PRODUCTID</td>
<td>=</td>
<td>msdyn_manufacturer.msdyn_manufacturer_id</td>
<td></td>
</tr>
<tr>
<td>MAINTENANCEASSETTYPEID</td>
<td>=</td>
<td>msdyn_maintenanceassettype.msdyn_maintenanceassettype_id</td>
<td></td>
</tr>
</tbody>
</table>

**Asset management warranty (msdyn_warranties)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>WARRANTYID</td>
<td>=</td>
<td>msdyn_warranty_id</td>
<td></td>
</tr>
</tbody>
</table>

**CDS Contacts V2 (contacts)**

This template synchronizes data between Finance and Operations apps and Dataverse.

Source filter: (AssociatedContactType = 1)

Reversed source filter: msdyn_contactforvendor eq true and msdyn_sellable eq false and msdyn_contactpersonid ne null
<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTACTPERSONPARTYNUMBER</td>
<td>=</td>
<td>msdyn_partyid.msdyn_partynumber</td>
<td></td>
</tr>
<tr>
<td>ASSOCIATEDCONTACTTYPE</td>
<td>&lt; &lt;</td>
<td>none</td>
<td>Vendor</td>
</tr>
<tr>
<td>ASSOCIATEDCONTACTNUMBER</td>
<td>=</td>
<td>msdyn_vendorcontactid.msdyn_vendoraccountnumber</td>
<td></td>
</tr>
<tr>
<td>EMPLOYMENTDEPARTMENT</td>
<td>=</td>
<td>department</td>
<td></td>
</tr>
<tr>
<td>NOTES</td>
<td>=</td>
<td>description</td>
<td></td>
</tr>
<tr>
<td>GOVERNMENTIDENTIFICATIONNUMBER</td>
<td>=</td>
<td>governmentid</td>
<td></td>
</tr>
<tr>
<td>ISRECEIVINGDIRECTMAIL</td>
<td>&gt; &lt;</td>
<td>donotemail</td>
<td></td>
</tr>
<tr>
<td>SPOUSENAME</td>
<td>=</td>
<td>spousesname</td>
<td></td>
</tr>
<tr>
<td>none</td>
<td>&gt; &gt;</td>
<td>msdyn_contactforvendor</td>
<td>True</td>
</tr>
<tr>
<td>CONTACTPERSONID</td>
<td>=</td>
<td>msdyn_contactpersonid</td>
<td></td>
</tr>
</tbody>
</table>

**CDS Exchange Rates (msdyn_currencyexchangerates)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCHANGERATETYPENAME</td>
<td>&gt;</td>
<td>msdyn_currencyexchangerateтип.msdyn_currencyexchangeratetypeid.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>FROMCURRENCYCODE</td>
<td>&gt;</td>
<td>msdyn_currencyexchangerateтип.msdyn_fromtransactioncurrencyid.isocurrencycode</td>
<td></td>
</tr>
<tr>
<td>TOCURRENCYCODE</td>
<td>&gt;</td>
<td>msdyn_currencyexchangerateтип.msdyn_totransactioncurrencyid.isocurrencycode</td>
<td></td>
</tr>
<tr>
<td>RATE</td>
<td>&gt;</td>
<td>msdyn_exchangerate</td>
<td></td>
</tr>
<tr>
<td>VALIDFROM</td>
<td>&gt;</td>
<td>msdyn_validfrom</td>
<td></td>
</tr>
<tr>
<td>VALIDTO</td>
<td>&gt;</td>
<td>msdyn_validto</td>
<td></td>
</tr>
</tbody>
</table>

**CDS Parties (msdyn_parties)**

This template synchronizes data between Finance and Operations apps and Dataverse.
<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARTYNUMBER</td>
<td>=</td>
<td>msdyn_partynumber</td>
<td></td>
</tr>
<tr>
<td>PARTYTYPE</td>
<td>&gt; &lt;</td>
<td>msdyn_partytype</td>
<td>msdyn_partytype</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAMEALIAS</td>
<td>=</td>
<td>msdyn_namealias</td>
<td></td>
</tr>
<tr>
<td>KNOWNAS</td>
<td>=</td>
<td>msdyn_nickname</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>LANGUAGEID</td>
<td>&gt; &lt;</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ar : 192350000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ar-ae : 192350001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>en-us : 192350015</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>en-gb : 192350000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>cs : 192350002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>da : 192350003</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>de : 192350004</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>de-at : 192350005</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>de-ch : 192350006</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>en-au : 192350007</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>en-ca : 192350008</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>en-ie : 192350010</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>en-in : 192350011</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>en-my : 192350012</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>en-nz : 192350013</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>en-sg : 192350014</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>en-za : 192350016</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>es : 192350017</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>es-mx : 192350018</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>et : 192350019</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>fi : 192350020</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>fr : 192350021</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>fr-be : 192350022</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>fr-ca : 192350023</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>fr-ch : 192350024</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>hu : 192350025</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>is : 192350026</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>it : 192350027</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>it-ch : 192350028</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ja : 192350029</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lt : 192350030</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lv : 192350031</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>nb-no : 192350032</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>nl : 192350033</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>nl-be : 192350034</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pl : 192350035</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pt-br : 192350036</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ru : 192350037</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sv : 192350038</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>th : 192350039</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tr : 192350040</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>zh-hans : 192350041</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| ORGANIZATIONNAME             | =         | msdyn_organizationname    |               |
| ORGANIZATIONABCCODE          | &gt; &lt;       | msdyn_organizationabccode |               |
|                               | a : 192350000 |                    |               |
|                               | b : 192350001 |                    |               |
|                               | c : 192350002 |                    |               |
|                               | none : 192350003 |                  |               |
| ORGANIZATIONNUMOFEmployees   | =         | msdyn_numberofemployees  |               |</p>
<table>
<thead>
<tr>
<th><strong>FINANCE AND OPERATIONS FIELD</strong></th>
<th><strong>MAP TYPE</strong></th>
<th><strong>CUSTOMER ENGAGEMENT FIELD</strong></th>
<th><strong>DEFAULT VALUE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>ORGANIZATIONNUMBER</td>
<td>=</td>
<td>msdyn_organizationnumber</td>
<td></td>
</tr>
<tr>
<td>ORGANIZATIONPHONETICNAME</td>
<td>=</td>
<td>msdyn_organizationphoneticname</td>
<td></td>
</tr>
<tr>
<td>PERSONFIRSTNAME</td>
<td>=</td>
<td>msdyn_firstname</td>
<td></td>
</tr>
<tr>
<td>PERSONMIDDLENAME</td>
<td>=</td>
<td>msdyn_middlename</td>
<td></td>
</tr>
<tr>
<td>PERSONLASTNAME</td>
<td>=</td>
<td>msdyn_lastname</td>
<td></td>
</tr>
<tr>
<td>PERSONLASTNAMEPREFIX</td>
<td>=</td>
<td>msdyn_lastnameprefix</td>
<td></td>
</tr>
<tr>
<td>PERSONINITIALS</td>
<td>=</td>
<td>msdyn_initials</td>
<td></td>
</tr>
<tr>
<td>PERSONPROFESSIONALTITLE</td>
<td>=</td>
<td>msdyn_professionaltitle</td>
<td></td>
</tr>
<tr>
<td>PERSONPROFESSIONALSUFFIX</td>
<td>=</td>
<td>msdyn_professionalsuffix</td>
<td></td>
</tr>
<tr>
<td>PERSONPHONETICFIRSTNAME</td>
<td>=</td>
<td>msdyn_phoneticfirstname</td>
<td></td>
</tr>
<tr>
<td>PERSONPHONETICLASTNAME</td>
<td>=</td>
<td>msdyn_phoneticlastname</td>
<td></td>
</tr>
<tr>
<td>PERSONPHONETICMIDDLENAME</td>
<td>=</td>
<td>msdyn_phoneticmiddlename</td>
<td></td>
</tr>
<tr>
<td>PERSONGENDER</td>
<td>&gt; &lt;</td>
<td>msdyn_gender</td>
<td></td>
</tr>
<tr>
<td></td>
<td>male : 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>female : 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>non-Specific : 192350000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>unknown : 192350001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PERSONMARITALSTATUS</td>
<td>&gt; &lt;</td>
<td>msdyn_maritalstatus</td>
<td></td>
</tr>
<tr>
<td></td>
<td>single : 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>married : 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>divorced : 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>widowhood : 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>none : 192350000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PERSONHOBBIES</td>
<td>=</td>
<td>msdyn_hobbies</td>
<td></td>
</tr>
<tr>
<td>PERSONCHILDRENNAMES</td>
<td>=</td>
<td>msdyn_childrennames</td>
<td></td>
</tr>
<tr>
<td>PERSONANNIVERSARYDAY</td>
<td>=</td>
<td>msdyn_anniversaryday</td>
<td></td>
</tr>
<tr>
<td>PERSONANNIVERSARYYEAR</td>
<td>=</td>
<td>msdyn_anniversaryyear</td>
<td></td>
</tr>
</tbody>
</table>
## PERSONBIRTHDAY

<table>
<thead>
<tr>
<th>Field</th>
<th>MAP TYPE</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERSONBIRTHDAY</td>
<td>=</td>
<td>msdyn_birthday</td>
<td></td>
</tr>
</tbody>
</table>

## PERSONBIRTHYEAR

<table>
<thead>
<tr>
<th>Field</th>
<th>MAP TYPE</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERSONBIRTHYEAR</td>
<td>=</td>
<td>msdyn_birthyear</td>
<td></td>
</tr>
</tbody>
</table>

## PERSONANNIVERSARYMONTH

<table>
<thead>
<tr>
<th>Field</th>
<th>MAP TYPE</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERSONANNIVERSARYMONTH</td>
<td>&gt; &lt;</td>
<td>msdyn_anniversarymonth</td>
<td></td>
</tr>
</tbody>
</table>

### PERSONANNIVERSARYMONTH Mapping

- January: 1
- February: 2
- March: 3
- April: 4
- May: 5
- June: 6
- July: 7
- August: 8
- September: 9
- October: 10
- November: 11
- December: 12

## PERSONBIRTHMONTH

<table>
<thead>
<tr>
<th>Field</th>
<th>MAP TYPE</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERSONBIRTHMONTH</td>
<td>&gt; &lt;</td>
<td>msdyn_birthmonth</td>
<td></td>
</tr>
</tbody>
</table>

### PERSONBIRTHMONTH Mapping

- None: 0
- January: 1
- February: 2
- March: 3
- April: 4
- May: 5
- June: 6
- July: 7
- August: 8
- September: 9
- October: 10
- November: 11
- December: 12

---

### CDS Party postal address locations (msdyn_partypostaladdresses)

This template synchronizes data between Finance and Operations apps and Dataverse.

## FINANCE AND OPERATIONS FIELD

<table>
<thead>
<tr>
<th>Field</th>
<th>MAP TYPE</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISPRIMARY</td>
<td>&gt; &lt;</td>
<td>msdyn_isprimary</td>
<td></td>
</tr>
</tbody>
</table>

### ISPRIMARY Mapping

- No: False
- Yes: True

## LOCATIONID

<table>
<thead>
<tr>
<th>Field</th>
<th>MAP TYPE</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCATIONID</td>
<td>=</td>
<td>msdyn_postaladdresscollectionid.msdyn_location</td>
<td></td>
</tr>
</tbody>
</table>

## PARTYNUMBER

<table>
<thead>
<tr>
<th>Field</th>
<th>MAP TYPE</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARTYNUMBER</td>
<td>=</td>
<td>msdyn_partyid.msdyn_partynumber</td>
<td></td>
</tr>
</tbody>
</table>

## PURPOSE

<table>
<thead>
<tr>
<th>Field</th>
<th>MAP TYPE</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PURPOSE</td>
<td>=</td>
<td>msdyn_postaladdresspurposes</td>
<td></td>
</tr>
<tr>
<td>Finance and Operations Field</td>
<td>Map Type</td>
<td>Customer Engagement Column</td>
<td>Default Value</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>IsLocationOwner</td>
<td>&gt;&gt;</td>
<td>msdyn_islocationowner</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td>True</td>
<td></td>
</tr>
</tbody>
</table>

**CDS inventory on-hand entries (msdyn_inventoryonhandentries)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RequestId</td>
<td>=</td>
<td>msdyn_request.msdyn_requestid</td>
<td></td>
</tr>
<tr>
<td>InventorySiteID</td>
<td>=</td>
<td>msdyn_inventorysite.msdyn_siteid</td>
<td></td>
</tr>
<tr>
<td>InventoryWarehouseID</td>
<td>=</td>
<td>msdyn_inventorywarehouse.msdyn_warehouseidentifier</td>
<td></td>
</tr>
<tr>
<td>AvailableOnHandQuantity</td>
<td>&gt;</td>
<td>msdyn_availableonhandquantity</td>
<td></td>
</tr>
<tr>
<td>AvailableOrderedQuantity</td>
<td>&gt;</td>
<td>msdyn_availableorderedquantity</td>
<td></td>
</tr>
<tr>
<td>OnHandQuantity</td>
<td>&gt;</td>
<td>msdyn_onhandquantity</td>
<td></td>
</tr>
<tr>
<td>OnOrderQuantity</td>
<td>&gt;</td>
<td>msdyn_onorderquantity</td>
<td></td>
</tr>
<tr>
<td>OrderedQuantity</td>
<td>&gt;</td>
<td>msdyn_orderedquantity</td>
<td></td>
</tr>
<tr>
<td>ReservedOnHandQuantity</td>
<td>&gt;</td>
<td>msdyn_reservedonhandquantity</td>
<td></td>
</tr>
<tr>
<td>ReservedOrderedQuantity</td>
<td>&gt;</td>
<td>msdyn_reservedorderedquantity</td>
<td></td>
</tr>
<tr>
<td>TotalAvailableQuantity</td>
<td>&gt;</td>
<td>msdyn_totalavailablequantity</td>
<td></td>
</tr>
<tr>
<td>AtDate</td>
<td>=</td>
<td>msdyn_atpdate</td>
<td></td>
</tr>
<tr>
<td>AtpQuantity</td>
<td>&gt;</td>
<td>msdyn_atpquantity</td>
<td></td>
</tr>
<tr>
<td>ProjectedIssueQuantity</td>
<td>&gt;</td>
<td>msdyn_projectedissuequantity</td>
<td></td>
</tr>
<tr>
<td>ProjectedOnHandQuantity</td>
<td>&gt;</td>
<td>msdyn_projectedonhandquantity</td>
<td></td>
</tr>
<tr>
<td>ProjectedReceiptQuantity</td>
<td>&gt;</td>
<td>msdyn_projectedreceiptquantity</td>
<td></td>
</tr>
<tr>
<td><strong>FINANCE AND OPERATIONS FIELD</strong></td>
<td><strong>MAP TYPE</strong></td>
<td><strong>CUSTOMER ENGAGEMENT COLUMN</strong></td>
<td><strong>DEFAULT VALUE</strong></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------</td>
<td>-------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>ORDERQUANTITY</td>
<td>&gt;</td>
<td>msdyn_orderquantity</td>
<td></td>
</tr>
<tr>
<td>UNAVAILABLEONHANDQUANTITY</td>
<td>&gt;</td>
<td>msdyn_unavailableonhandquantity</td>
<td></td>
</tr>
</tbody>
</table>

**CDS inventory on-hand requests (msdyn_inventoryonhandrequests)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th><strong>FINANCE AND OPERATIONS FIELD</strong></th>
<th><strong>MAP TYPE</strong></th>
<th><strong>CUSTOMER ENGAGEMENT COLUMN</strong></th>
<th><strong>DEFAULT VALUE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>REQUESTID</td>
<td>=</td>
<td>msdyn_requestid</td>
<td></td>
</tr>
<tr>
<td>PRODUCTNUMBER</td>
<td>&lt;</td>
<td>msdyn_product.msdyn_productnumber</td>
<td></td>
</tr>
<tr>
<td>ISATPCALCULATION</td>
<td>&lt; &lt;</td>
<td>msdyn_isatpcalculation</td>
<td></td>
</tr>
<tr>
<td>ORDERQUANTITY</td>
<td>&lt;</td>
<td>msdyn_orderquantity</td>
<td></td>
</tr>
<tr>
<td>INVENTORIESITEID</td>
<td>&lt;</td>
<td>msdyn_inventorysite.msdyn_siteid</td>
<td></td>
</tr>
<tr>
<td>INVENTORYWAREHOUSEID</td>
<td>&lt;</td>
<td>msdyn_inventorywarehouse.msdyn_warehouseidentifier</td>
<td></td>
</tr>
<tr>
<td>REFERENCENUMBER</td>
<td>&lt;</td>
<td>msdyn_referencenumber</td>
<td></td>
</tr>
<tr>
<td>LINECREATIONSEQUENCENUMBER</td>
<td>&lt;</td>
<td>msdyn_linecreationsequencenumber</td>
<td></td>
</tr>
</tbody>
</table>

**CDS postal address history V2 (msdyn_postaladdresses)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th><strong>FINANCE AND OPERATIONS FIELD</strong></th>
<th><strong>MAP TYPE</strong></th>
<th><strong>CUSTOMER ENGAGEMENT COLUMN</strong></th>
<th><strong>DEFAULT VALUE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCATIONID</td>
<td>=</td>
<td>msdyn_postaladdresscollectionid.msdyn_location</td>
<td></td>
</tr>
<tr>
<td>VALIDFROM</td>
<td>=</td>
<td>msdyn_validfrom</td>
<td></td>
</tr>
<tr>
<td>VALIDTO</td>
<td>=</td>
<td>msdyn_validto</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>STREET</td>
<td>=</td>
<td>msdyn_street</td>
<td></td>
</tr>
<tr>
<td>STREETNUMBER</td>
<td>=</td>
<td>msdyn_streectnumber</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>CITY</td>
<td>=</td>
<td>msdyn_city</td>
<td></td>
</tr>
<tr>
<td>STATE</td>
<td>=</td>
<td>msdyn_state</td>
<td></td>
</tr>
<tr>
<td>COUNTRYREGIONID</td>
<td>=</td>
<td>msdyn_countryregionid</td>
<td></td>
</tr>
<tr>
<td>ZIPCODE</td>
<td>=</td>
<td>msdyn_zipcode</td>
<td></td>
</tr>
<tr>
<td>COUNTY</td>
<td>=</td>
<td>msdyn_county</td>
<td></td>
</tr>
<tr>
<td>DISTRICTNAME</td>
<td>=</td>
<td>msdyn_district</td>
<td></td>
</tr>
<tr>
<td>BUILDINGCOMPLIMENT</td>
<td>=</td>
<td>msdyn_buildingcompliment</td>
<td></td>
</tr>
<tr>
<td>POSTBOX</td>
<td>=</td>
<td>msdyn_postbox</td>
<td></td>
</tr>
</tbody>
</table>

**CDS postal address locations (msdyn_postaladdresscollections)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
<tr>
<td>LOCATIONID</td>
<td>=</td>
<td>msdyn_location</td>
<td></td>
</tr>
</tbody>
</table>

**CDS purchase order line entity (msdyn_purchaseorderproducts)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQUESTEDDELIVERYDATE</td>
<td>=</td>
<td>msdyn_dateexpected</td>
<td></td>
</tr>
<tr>
<td>LINEDESCRIPTION</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
<tr>
<td>LINENUMBER</td>
<td>=</td>
<td>msdyn_lineorder</td>
<td></td>
</tr>
<tr>
<td>PRODUCTNUMBER</td>
<td>=</td>
<td>msdyn_product.msdyn_productnumber</td>
<td></td>
</tr>
<tr>
<td>PURCHASEORDERNUMBER</td>
<td>=</td>
<td>msdyn_purchaseordernumber.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>ORDEREDPURCHASEQUANTITY</td>
<td>=</td>
<td>msdyn_quantity</td>
<td></td>
</tr>
<tr>
<td>LINEAMOUNT</td>
<td>&gt;</td>
<td>msdyn_lineamount</td>
<td></td>
</tr>
<tr>
<td>PURCHASEPRICE</td>
<td>&gt;</td>
<td>msdyn_unitcost</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT FIELD</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>---------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>BARCODE</td>
<td>=</td>
<td>msdyn_barcode</td>
<td></td>
</tr>
<tr>
<td>CATCHWEIGHTUNITSYMBO L</td>
<td>=</td>
<td>msdyn_catchweightunitsym bol.msdyn_symbol</td>
<td></td>
</tr>
<tr>
<td>CONFIRMEDSHIPPINGDATE</td>
<td>=</td>
<td>msdyn_confirmedshippingdate</td>
<td></td>
</tr>
<tr>
<td>CUSTOMERREFERENCE</td>
<td>=</td>
<td>msdyn_customerreference</td>
<td></td>
</tr>
<tr>
<td>CUSTOMERREQUISITIONNUMBER</td>
<td>=</td>
<td>msdyn_customerrequisition number</td>
<td></td>
</tr>
<tr>
<td>EXTERNALITEMNUMBER</td>
<td>=</td>
<td>msdyn_externalitemnumber</td>
<td></td>
</tr>
<tr>
<td>ISPARTIALDELIVERYPREVENTED</td>
<td>&gt; &lt;</td>
<td>msdyn_ispartialdeliveryprevented</td>
<td></td>
</tr>
<tr>
<td>LINEDISCOUNTAMOUNT</td>
<td>&gt;</td>
<td>msdyn_linediscountamount</td>
<td></td>
</tr>
<tr>
<td>LINEDISCOUNTPERCENTAGE</td>
<td>&gt;</td>
<td>msdyn_linediscountpercentage</td>
<td></td>
</tr>
<tr>
<td>ORDEREDCATCHWEIGHTQUANTITY</td>
<td>=</td>
<td>msdyn_orderedcatchweightquantity</td>
<td></td>
</tr>
<tr>
<td>PURCHASEORDERLINENSTATUS</td>
<td>&gt;&gt;</td>
<td>msdyn_purchaseorderlinestatus</td>
<td></td>
</tr>
<tr>
<td>PURCHASEPRICEQUANTITY</td>
<td>=</td>
<td>msdyn_purchasepricequantity</td>
<td></td>
</tr>
<tr>
<td>RECEIVINGSITEID</td>
<td>=</td>
<td>msdyn_receivingssiteid.msdyn_siteid</td>
<td></td>
</tr>
<tr>
<td>REQUESTEDSHIPPINGDATE</td>
<td>=</td>
<td>msdyn_requestedshippingdate</td>
<td></td>
</tr>
<tr>
<td>REQUESTERPERSONNELNUMBER</td>
<td>=</td>
<td>msdyn_requesterpersonnel number.cdm_workernumber</td>
<td></td>
</tr>
<tr>
<td>PROCUREMENTPRODUCTCATEGORYNAME</td>
<td>&gt;</td>
<td>msdyn_procurementproductcategory.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>PROCUREMENTPRODUCTCATEGOR YHIERACHYNAME</td>
<td>&gt;</td>
<td>msdyn_procurementproductcategory.msdyn_hierarchy.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>CURRENCYCODE</td>
<td>&gt;</td>
<td>transactioncurrencyid.isocurrencycode</td>
<td></td>
</tr>
<tr>
<td>CONFIRMEDDELIVERYDATE</td>
<td>=</td>
<td>msdyn_confirmeddeliverydate</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESCITY</td>
<td>&gt;</td>
<td>msdyn_deliveryaddresscity</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSTRNUNBER</td>
<td>&gt;</td>
<td>msdyn_deliveryaddressstreetnumber</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSZIPCODE</td>
<td>&gt;</td>
<td>msdyn_deliveryaddresszipcode</td>
<td></td>
</tr>
<tr>
<td>FORMATTEDDELIVERYADDRESS</td>
<td>&gt;</td>
<td>msdyn_formatteddeliveryaddress</td>
<td></td>
</tr>
<tr>
<td>PURCHASEUNITSYMBOL</td>
<td>=</td>
<td>msdyn_unit.msdyn_symbol</td>
<td></td>
</tr>
<tr>
<td>RECEIVINGWAREHOUSEID</td>
<td>=</td>
<td>msdyn_associatetowardsme.msdyn_warehouseidentification</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSDESCRIPTION</td>
<td>&gt;</td>
<td>msdyn_deliveryaddressdescription</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSNAME</td>
<td>&gt;</td>
<td>msdyn_deliveryaddressname</td>
<td></td>
</tr>
</tbody>
</table>

**CDS purchase order line soft deleted entity (msdyn_purchaseorderproducts)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINENUMBER</td>
<td>&gt;</td>
<td>msdyn_lineorder</td>
<td></td>
</tr>
<tr>
<td>PURCHASEORDERNUMBER</td>
<td>&gt;</td>
<td>msdyn_purchaseordermsdyn_name</td>
<td></td>
</tr>
<tr>
<td>ISDELETED</td>
<td>&gt;&gt;</td>
<td>msdyn_issoftdeletedinscm</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>yes</th>
<th>true</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>false</td>
</tr>
</tbody>
</table>

**CDS released distinct products (products)**
This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCTNUMBER</td>
<td>&gt;</td>
<td>msdyn_productnumber</td>
<td></td>
</tr>
<tr>
<td>PRODUCTNAME</td>
<td>&gt;</td>
<td>name</td>
<td></td>
</tr>
<tr>
<td>PRODUCTDESCRIPTION</td>
<td>&gt;</td>
<td>description</td>
<td></td>
</tr>
<tr>
<td>ITEMNUMBER</td>
<td>&gt;</td>
<td>msdyn_itemnumber</td>
<td></td>
</tr>
<tr>
<td>CURRENCYCODE</td>
<td>&gt;</td>
<td>transactioncurrencyid.isocurrencycode</td>
<td></td>
</tr>
<tr>
<td>SALESUNITSYMBOL</td>
<td>&gt;</td>
<td>defaultuomid.msdyn_symbol</td>
<td></td>
</tr>
<tr>
<td>SALESPRICE</td>
<td>&gt;</td>
<td>price</td>
<td></td>
</tr>
<tr>
<td>UNITCOST</td>
<td>&gt;</td>
<td>currentcost</td>
<td></td>
</tr>
<tr>
<td>PRODUCTTYPE</td>
<td>&gt;&gt;</td>
<td>producttypecode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>item</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>service</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>SALESUNITDECIMALPRECISION</td>
<td>&gt;&gt;</td>
<td>quantitydecimal</td>
<td>0</td>
</tr>
<tr>
<td>ISCATCHWEIGHTPRODUCT</td>
<td>&gt;&gt;</td>
<td>msdyn_iscatchweight</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>ISSTOCKEDPRODUCT</td>
<td>&gt;&gt;</td>
<td>msdyn_isstockedproduct</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>PRODUCTCOLORID</td>
<td>&gt;</td>
<td>msdyn_productcolor.msdyn_productcolorname</td>
<td></td>
</tr>
<tr>
<td>PRODUCTCONFIGURATIONID</td>
<td>&gt;</td>
<td>msdyn_productconfiguration.msdyn_productconfiguration</td>
<td></td>
</tr>
<tr>
<td>PRODUCTSIZEID</td>
<td>&gt;</td>
<td>msdyn_productsize.msdyn_productsize</td>
<td></td>
</tr>
<tr>
<td>PRODUCTSTYLEID</td>
<td>&gt;</td>
<td>msdyn_productstyle.msdyn_productstyle</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>FIELDSERVICEPRODUCTTYPE</td>
<td>&gt;&gt;</td>
<td>msdyn_fieldserviceproducttype</td>
<td></td>
</tr>
<tr>
<td>Inventory</td>
<td>690970000</td>
<td>msdyn_fieldserviceproducttype</td>
<td></td>
</tr>
<tr>
<td>NonInventory</td>
<td>690970001</td>
<td>msdyn_fieldserviceproducttype</td>
<td></td>
</tr>
<tr>
<td>Service</td>
<td>690970002</td>
<td>msdyn_fieldserviceproducttype</td>
<td></td>
</tr>
</tbody>
</table>

### CDS sales order headers (salesorders)

This template synchronizes data between Finance and Operations apps and Dataverse.

Reversed source filter: msdyn_ordertype eq 192350000

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SALESORDERNUMBER</td>
<td>=</td>
<td>msdyn_salesordernumber</td>
<td></td>
</tr>
<tr>
<td>ORDERINGCUSTOMERACCOUNTNUMBER</td>
<td>=</td>
<td>customerid.Account(accounumber).Contact(msdyn_contactpersonid)</td>
<td></td>
</tr>
<tr>
<td>CURRENCYCODE</td>
<td>=</td>
<td>transactioncurrencyid.isocurrencycode</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSCITY</td>
<td>=</td>
<td>shipto_city</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSCOUNTRYREGIONISOCODE</td>
<td>&gt;&gt;</td>
<td>shipto_country</td>
<td></td>
</tr>
<tr>
<td>us</td>
<td>united states</td>
<td></td>
<td></td>
</tr>
<tr>
<td>de</td>
<td>germany</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fr</td>
<td>france</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gb</td>
<td>united kingdom</td>
<td></td>
<td></td>
</tr>
<tr>
<td>jp</td>
<td>japan</td>
<td></td>
<td></td>
</tr>
<tr>
<td>in</td>
<td>india</td>
<td></td>
<td></td>
</tr>
<tr>
<td>es</td>
<td>spain</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nz</td>
<td>new zealand</td>
<td></td>
<td></td>
</tr>
<tr>
<td>au</td>
<td>australia</td>
<td></td>
<td></td>
</tr>
<tr>
<td>co</td>
<td>colombia</td>
<td></td>
<td></td>
</tr>
<tr>
<td>za</td>
<td>south africa</td>
<td></td>
<td></td>
</tr>
<tr>
<td>be</td>
<td>belgium</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ca</td>
<td>canada</td>
<td></td>
<td></td>
</tr>
<tr>
<td>at</td>
<td>austria</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tr</td>
<td>turkey</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cn</td>
<td>china</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dk</td>
<td>denmark</td>
<td></td>
<td></td>
</tr>
<tr>
<td>se</td>
<td>sweden</td>
<td></td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>norway</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fi</td>
<td>finland</td>
<td></td>
<td></td>
</tr>
<tr>
<td>eg</td>
<td>egypt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSTYPENUMBER</td>
<td>=</td>
<td>shipto_line2</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSZIPCODE</td>
<td>=</td>
<td>shipto_postalcode</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT FIELD</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------</td>
<td>---------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>DELIVERYADDRESSSTREET</td>
<td>=</td>
<td>shipto_line1</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSSTATEID</td>
<td>=</td>
<td>shipto_stateorprovince</td>
<td></td>
</tr>
<tr>
<td>SALESDORDERNAME</td>
<td>&gt;</td>
<td>name</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSCITY</td>
<td>&gt;</td>
<td>billto_city</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSSTREET</td>
<td>&gt;</td>
<td>billto_line1</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSSTREETNUMBER</td>
<td>&gt;</td>
<td>billto_line2</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSCOUNTRYREGIONISOCODE</td>
<td>&gt;</td>
<td>billto_country</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSSTATEID</td>
<td>&gt;</td>
<td>billto_stateorprovince</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSZIPCODE</td>
<td>&gt;</td>
<td>billto_postalcode</td>
<td></td>
</tr>
<tr>
<td>ORDERTOTALAMOUNT</td>
<td>&gt;</td>
<td>totalamount</td>
<td></td>
</tr>
<tr>
<td>TOTALDISCOUNTAMOUNT</td>
<td>&gt;</td>
<td>discountamount</td>
<td></td>
</tr>
<tr>
<td>ORDERTOTALTAXAMOUNT</td>
<td>&gt;</td>
<td>totaltax</td>
<td></td>
</tr>
<tr>
<td>ORDERTOTALCHARGESAMOUNT</td>
<td>&gt;</td>
<td>freightamount</td>
<td></td>
</tr>
<tr>
<td>AREPRICESINCLUDINGSALESTAX</td>
<td>&lt;</td>
<td>msdyn_arepricesincludingsales</td>
<td>msdyn_arepricesincludingsales</td>
</tr>
<tr>
<td>CONFIRMEDRECEIPTDATE</td>
<td>=</td>
<td>msdyn_confirmedreceiptdate</td>
<td></td>
</tr>
<tr>
<td>CONFIRMEDSHIPPINGDATE</td>
<td>=</td>
<td>msdyn_confirmedshippingdate</td>
<td></td>
</tr>
<tr>
<td>CONTACTPERSONID</td>
<td>=</td>
<td>msdyn_associatedcontact.msdyn_contactforpartynumber</td>
<td></td>
</tr>
<tr>
<td>CUSTOMERREQUESTIONNUMBER</td>
<td>=</td>
<td>msdyn_customerrequisitionnumber</td>
<td></td>
</tr>
<tr>
<td>DEFAULTSHIPPINGSITEID</td>
<td>=</td>
<td>msdyn_defaultshippingsite/msdyn_siteid</td>
<td></td>
</tr>
<tr>
<td>DEFAULT SHIPPING WAREHOUSEID</td>
<td>=</td>
<td>msdyn_defaultshippingwarehouse/msdyn_warehouseidentifier</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT FIELD</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>---------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>DELIVERYADDRESSCOUNTYID</td>
<td>=</td>
<td>msdyn_deliveryaddresscoun tyid</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSDESCRIPTION</td>
<td>=</td>
<td>msdyn_deliveryaddressdescr iption</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSDISTRICTNAME</td>
<td>=</td>
<td>msdyn_deliveryaddressdistri ctname</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSDUNSNUMBER</td>
<td>=</td>
<td>msdyn_deliveryaddressdunsn umber</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSLATITUDE</td>
<td>=</td>
<td>msdyn_deliveryaddresslatitu de</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSLOCATIONID</td>
<td>=</td>
<td>msdyn_deliveryaddresslocationid</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSLONGITUDE</td>
<td>=</td>
<td>msdyn_deliveryaddresslongitu de</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSNAME</td>
<td>=</td>
<td>msdyn_deliveryaddressname</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSPOSTBOX</td>
<td>=</td>
<td>msdyn_deliveryaddresspostbox</td>
<td></td>
</tr>
<tr>
<td>DELIVERYBUILDINGCOMPLIMENT</td>
<td>=</td>
<td>msdyn_deliverybuildingcompliment</td>
<td></td>
</tr>
<tr>
<td>FORMATTEDDELIVERYADDRESS</td>
<td>&gt;</td>
<td>msdyn_formatteddeliveryad dress</td>
<td></td>
</tr>
<tr>
<td>EMAIL</td>
<td>=</td>
<td>emailaddress</td>
<td></td>
</tr>
<tr>
<td>FORMATTEDINVOICEADDRESS</td>
<td>&gt;</td>
<td>msdyn_formattedinvoiceadress</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSCOUNTYID</td>
<td>&gt;</td>
<td>msdyn_invoiceaddresscount yid</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSDISTRICTNAME</td>
<td>&gt;</td>
<td>msdyn_invoiceaddressdistrictname</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSLATITUDE</td>
<td>&gt;</td>
<td>msdyn_invoiceaddresslatitud e</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSLONGITUDE</td>
<td>&gt;</td>
<td>msdyn_invoiceaddresslongitu de</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSPOSTBOX</td>
<td>&gt;</td>
<td>msdyn_invoiceaddresspostbox</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>INVOICEBUILDINGCOMPLIMENT</td>
<td>&gt;</td>
<td>msdyn_invoicebuildingcomp</td>
<td></td>
</tr>
<tr>
<td>INVOICECUSTOMERACCOUNTNUMBER</td>
<td>=</td>
<td>msdyn_invoicecustomerid.A</td>
<td></td>
</tr>
<tr>
<td>ISDELIVERYADDRESSORDERSPECIFIC</td>
<td>&gt; &lt;</td>
<td>msdyn_isdeliveryaddressord</td>
<td></td>
</tr>
<tr>
<td>ISDELIVERYADDRESSPRIVATE</td>
<td>&gt; &lt;</td>
<td>msdyn_isdeliveryaddresspri</td>
<td></td>
</tr>
<tr>
<td>ISINVOICEADDRESSPRIVATE</td>
<td>&gt; &gt;</td>
<td>msdyn_isinvoiceaddresspriv</td>
<td></td>
</tr>
<tr>
<td>ISONETIMECUSTOMER</td>
<td>&gt; &lt;</td>
<td>msdyn_isonetimecustomer</td>
<td></td>
</tr>
<tr>
<td>ISSALESPROCESSINGSTOPPED</td>
<td>&gt; &lt;</td>
<td>msdyn_issalesprocessingsto</td>
<td></td>
</tr>
<tr>
<td>PAYMENTTERMSBASEDATE</td>
<td>=</td>
<td>msdyn_paymenttermsbasedate</td>
<td></td>
</tr>
<tr>
<td>PAYMENTTERMSNAME</td>
<td>=</td>
<td>msdyn_paymentterms.msdy</td>
<td></td>
</tr>
<tr>
<td>PRICECUSTOMERGROUPCODE</td>
<td>=</td>
<td>msdyn_pricecustomergroup</td>
<td></td>
</tr>
<tr>
<td>QUOTATIONNUMBER</td>
<td>=</td>
<td>msdyn_quotationnumber</td>
<td></td>
</tr>
<tr>
<td>REQUESTEDRECEIPTDATE</td>
<td>=</td>
<td>msdynRequestedreceiptdat</td>
<td></td>
</tr>
<tr>
<td>REQUESTEDSHIPPINGDATE</td>
<td>=</td>
<td>requestdeliveryby</td>
<td></td>
</tr>
<tr>
<td>SALESORDERPROMISINGMETHOD</td>
<td>&gt; &lt;</td>
<td>msdyn_salesorderpromising</td>
<td></td>
</tr>
<tr>
<td></td>
<td>none : 192350000</td>
<td>salesLeadTime : 192350001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>atp : 192350002</td>
<td>attp : 192350003</td>
<td></td>
</tr>
<tr>
<td></td>
<td>atpPlusIssueMargin : 192350004</td>
<td>ctp : 192350004</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------</td>
<td>-----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>URL</td>
<td>=</td>
<td>msdyn_url</td>
<td></td>
</tr>
<tr>
<td>SALESORDERPROCESSINGSTATUS</td>
<td>&lt;</td>
<td>msdyn_processingstatus</td>
<td></td>
</tr>
<tr>
<td></td>
<td>active</td>
<td>192350000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>confirmed</td>
<td>192350001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>picked</td>
<td>192350002</td>
<td></td>
</tr>
<tr>
<td></td>
<td>partiallyDelivered</td>
<td>192350003</td>
<td></td>
</tr>
<tr>
<td></td>
<td>delivered</td>
<td>192350004</td>
<td></td>
</tr>
<tr>
<td></td>
<td>invoiced</td>
<td>192350005</td>
<td></td>
</tr>
<tr>
<td></td>
<td>partiallyInvoiced</td>
<td>192350006</td>
<td></td>
</tr>
<tr>
<td></td>
<td>canceled</td>
<td>192350007</td>
<td></td>
</tr>
<tr>
<td><strong>FIELD</strong></td>
<td><strong>MAP TYPE</strong></td>
<td><strong>CUSTOMER ENGAGEMENT COLUMN</strong></td>
<td><strong>DEFAULT VALUE</strong></td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------</td>
<td>--------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>LANGUAGEID</td>
<td><code>&lt;</code></td>
<td></td>
<td>msdyn_language</td>
</tr>
<tr>
<td></td>
<td><code>ar</code> :</td>
<td></td>
<td>192350000</td>
</tr>
<tr>
<td></td>
<td><code>ar-ae</code> :</td>
<td></td>
<td>192350001</td>
</tr>
<tr>
<td></td>
<td><code>cs</code> :</td>
<td></td>
<td>192350002</td>
</tr>
<tr>
<td></td>
<td><code>da</code> :</td>
<td></td>
<td>192350003</td>
</tr>
<tr>
<td></td>
<td><code>de</code> :</td>
<td></td>
<td>192350004</td>
</tr>
<tr>
<td></td>
<td><code>de-at</code> :</td>
<td></td>
<td>192350005</td>
</tr>
<tr>
<td></td>
<td><code>de-ch</code> :</td>
<td></td>
<td>192350006</td>
</tr>
<tr>
<td></td>
<td><code>en-au</code> :</td>
<td></td>
<td>192350007</td>
</tr>
<tr>
<td></td>
<td><code>en-gb</code> :</td>
<td></td>
<td>192350009</td>
</tr>
<tr>
<td></td>
<td><code>en-ie</code> :</td>
<td></td>
<td>192350010</td>
</tr>
<tr>
<td></td>
<td><code>en-in</code> :</td>
<td></td>
<td>192350011</td>
</tr>
<tr>
<td></td>
<td><code>en-ca</code> :</td>
<td></td>
<td>192350008</td>
</tr>
<tr>
<td></td>
<td><code>en-my</code> :</td>
<td></td>
<td>192350012</td>
</tr>
<tr>
<td></td>
<td><code>en-nz</code> :</td>
<td></td>
<td>192350013</td>
</tr>
<tr>
<td></td>
<td><code>en-sg</code> :</td>
<td></td>
<td>192350014</td>
</tr>
<tr>
<td></td>
<td><code>en-us</code> :</td>
<td></td>
<td>192350015</td>
</tr>
<tr>
<td></td>
<td><code>en-za</code> :</td>
<td></td>
<td>192350016</td>
</tr>
<tr>
<td></td>
<td><code>es</code> :</td>
<td></td>
<td>192350017</td>
</tr>
<tr>
<td></td>
<td><code>es-mx</code> :</td>
<td></td>
<td>192350018</td>
</tr>
<tr>
<td></td>
<td><code>et</code> :</td>
<td></td>
<td>192350019</td>
</tr>
<tr>
<td></td>
<td><code>fi</code> :</td>
<td></td>
<td>192350020</td>
</tr>
<tr>
<td></td>
<td><code>fr</code> :</td>
<td></td>
<td>192350021</td>
</tr>
<tr>
<td></td>
<td><code>fr-be</code> :</td>
<td></td>
<td>192350022</td>
</tr>
<tr>
<td></td>
<td><code>fr-ca</code> :</td>
<td></td>
<td>192350023</td>
</tr>
<tr>
<td></td>
<td><code>fr-ch</code> :</td>
<td></td>
<td>192350024</td>
</tr>
<tr>
<td></td>
<td><code>hu</code> :</td>
<td></td>
<td>192350025</td>
</tr>
<tr>
<td></td>
<td><code>is</code> :</td>
<td></td>
<td>192350026</td>
</tr>
<tr>
<td></td>
<td><code>it</code> :</td>
<td></td>
<td>192350027</td>
</tr>
<tr>
<td></td>
<td><code>it-ch</code> :</td>
<td></td>
<td>192350028</td>
</tr>
<tr>
<td></td>
<td><code>ja</code> :</td>
<td></td>
<td>192350029</td>
</tr>
<tr>
<td></td>
<td><code>lt</code> :</td>
<td></td>
<td>192350030</td>
</tr>
<tr>
<td></td>
<td><code>lv</code> :</td>
<td></td>
<td>192350031</td>
</tr>
<tr>
<td></td>
<td><code>nb-no</code> :</td>
<td></td>
<td>192350032</td>
</tr>
<tr>
<td></td>
<td><code>nl</code> :</td>
<td></td>
<td>192350033</td>
</tr>
<tr>
<td></td>
<td><code>nl-be</code> :</td>
<td></td>
<td>192350034</td>
</tr>
<tr>
<td></td>
<td><code>pl</code> :</td>
<td></td>
<td>192350035</td>
</tr>
<tr>
<td></td>
<td><code>pt-br</code> :</td>
<td></td>
<td>192350036</td>
</tr>
<tr>
<td></td>
<td><code>ru</code> :</td>
<td></td>
<td>192350037</td>
</tr>
<tr>
<td></td>
<td><code>sv</code> :</td>
<td></td>
<td>192350038</td>
</tr>
<tr>
<td></td>
<td><code>th</code> :</td>
<td></td>
<td>192350039</td>
</tr>
<tr>
<td></td>
<td><code>tr</code> :</td>
<td></td>
<td>192350040</td>
</tr>
<tr>
<td></td>
<td><code>zh-hans</code> :</td>
<td></td>
<td>192350041</td>
</tr>
<tr>
<td>CUSTOMERSORDERREFERENCE</td>
<td><code>=</code></td>
<td></td>
<td>msdyn_customersorderreference</td>
</tr>
<tr>
<td>DELIVERYMODECODE</td>
<td><code>=</code></td>
<td></td>
<td>msdyn_deliverymode.msdyn_name</td>
</tr>
<tr>
<td>DELIVERYTERMSCODE</td>
<td><code>=</code></td>
<td></td>
<td>msdyn_deliveryterms.msdyn_termcode</td>
</tr>
<tr>
<td><strong>FINANCE AND OPERATIONS FIELD</strong></td>
<td><strong>MAP TYPE</strong></td>
<td><strong>CUSTOMER ENGAGEMENT FIELD</strong></td>
<td><strong>DEFAULT VALUE</strong></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------</td>
<td>-------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td><strong>SALESORDERORIGINCODE</strong></td>
<td>=</td>
<td>msdyn_salesorderorigin.ms</td>
<td>dyn_origincode</td>
</tr>
<tr>
<td>None</td>
<td>&gt;&gt;</td>
<td>msdyn_ordertype</td>
<td>192350000</td>
</tr>
<tr>
<td><strong>CURRENCYCODE</strong></td>
<td>&gt;</td>
<td>msdyn_isocurrencycode</td>
<td></td>
</tr>
</tbody>
</table>

**CDS sales order lines (salesorderdetails)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th><strong>FINANCE AND OPERATIONS FIELD</strong></th>
<th><strong>MAP TYPE</strong></th>
<th><strong>CUSTOMER ENGAGEMENT FIELD</strong></th>
<th><strong>DEFAULT VALUE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CURRENCYCODE</strong></td>
<td>&gt;</td>
<td>transactioncurrencyid.isocurrencycodem</td>
<td></td>
</tr>
<tr>
<td><strong>DELIVERYADDRESSCITY</strong></td>
<td>=</td>
<td>shipto_city</td>
<td></td>
</tr>
<tr>
<td><strong>DELIVERYADDRESSCOUNTRYREGIONISOCODE</strong></td>
<td>&gt;&gt;</td>
<td>shipto_country</td>
<td></td>
</tr>
<tr>
<td>us: united states</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>de: germany</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fr: france</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gb: united kingdom</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>jp: japan</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>in: india</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>es: spain</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nz: new zealand</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>au: australia</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>co: colombia</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>za: south africa</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>be: belgium</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ca: canada</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>at: austria</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tr: turkey</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cn: china</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dk: denmark</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>se: sweden</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>no: norway</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fi: finland</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eg: egypt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>DELIVERYADDRESSZIPCODE</strong></td>
<td>=</td>
<td>shipto_postalcode</td>
<td></td>
</tr>
<tr>
<td><strong>DELIVERYADDRESSSTATEID</strong></td>
<td>=</td>
<td>shipto_stateorprovince</td>
<td></td>
</tr>
<tr>
<td><strong>DELIVERYADDRESSSTREET</strong></td>
<td>=</td>
<td>shipto_line1</td>
<td></td>
</tr>
<tr>
<td><strong>DELIVERYADDRESSSTREETNUMBER</strong></td>
<td>=</td>
<td>shipto_line2</td>
<td></td>
</tr>
<tr>
<td><strong>LINEAMOUNT</strong></td>
<td>&gt;</td>
<td>extendedamount</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT FIELD</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------</td>
<td>---------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>LINECREATIONSEQUENCENUMBER</td>
<td>=</td>
<td>sequencenumber</td>
<td></td>
</tr>
<tr>
<td>ORDEREDSALESCOUNTERTY</td>
<td>=</td>
<td>quantity</td>
<td></td>
</tr>
<tr>
<td>PRODUCTNAME</td>
<td>=</td>
<td>description</td>
<td></td>
</tr>
<tr>
<td>PRODUCTNUMBER</td>
<td>=</td>
<td>productid.msdyn_productnumber</td>
<td></td>
</tr>
<tr>
<td>SALESORDERNUMBER</td>
<td>=</td>
<td>salesorderid.msdyn_salesordernumber</td>
<td></td>
</tr>
<tr>
<td>SALESUNITSYMBOL</td>
<td>=</td>
<td>uomid.msdyn_symbol</td>
<td></td>
</tr>
<tr>
<td>TOTALDISCOUNTAMOUNT</td>
<td>=</td>
<td>manualdiscountamount</td>
<td></td>
</tr>
<tr>
<td>TOTALTAXAMOUNT</td>
<td>&gt;</td>
<td>tax</td>
<td></td>
</tr>
<tr>
<td>SALESPRICE</td>
<td>=</td>
<td>priceperunit</td>
<td></td>
</tr>
<tr>
<td>LINEAMOUNT</td>
<td>&gt;</td>
<td>baseamount</td>
<td></td>
</tr>
<tr>
<td>SALESPRODUCTCATEGORYNAME</td>
<td>=</td>
<td>msdyn_salesproductcategoryname</td>
<td></td>
</tr>
<tr>
<td>SALESPRODUCTCATEGORYHIERARCHYNAME</td>
<td>&gt;</td>
<td>msdyn_salesproductcategoryname.msdyn_hierarchy.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>ISDELIVERYADDRESSORDERSPECIFIC</td>
<td>&gt; &lt;</td>
<td>msdyn_isdeliveryaddressspecific</td>
<td></td>
</tr>
<tr>
<td>ISDELIVERYADDRESSPRIVATE</td>
<td>&gt; &lt;</td>
<td>msdyn_isdeliveryaddressprivate</td>
<td></td>
</tr>
<tr>
<td>ISLINESTOPPED</td>
<td>&gt; &lt;</td>
<td>msdyn_islinestopped</td>
<td></td>
</tr>
<tr>
<td>ALLOWEDOVERTOTALDELIVERYPERCENTAGE</td>
<td>=</td>
<td>msdyn_allowedovertotaldeliverypercentage</td>
<td></td>
</tr>
<tr>
<td>ALLOWEDUNDERDELIVERYPERCENTAGE</td>
<td>=</td>
<td>msdyn_allowedunderdeliverypercentage</td>
<td></td>
</tr>
<tr>
<td>CONFIRMEDSHIPPINGDATE</td>
<td>=</td>
<td>msdyn_confirmedshippingdate</td>
<td></td>
</tr>
</tbody>
</table>

---

**FINANCE AND OPERATIONS FIELD**
- **MAP TYPE**
- **CUSTOMER ENGAGEMENT COLUMN**
- **DEFAULT VALUE**

**CUSTOMER ENGAGEMENT FIELD**
- **MAP TYPE**
- **COLUMN**
- **DEFAULT VALUE**
<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIRMEDRECEIPTDATE</td>
<td>=</td>
<td>msdyn_confirmedreceiptdate</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSCOUNTYID</td>
<td>=</td>
<td>msdyn_deliveryaddresscountyid</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSDESCRIPTION</td>
<td>=</td>
<td>msdyn_deliveryaddressdescription</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSDISTRICTNAME</td>
<td>=</td>
<td>msdyn_deliveryaddressdistrictname</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSDUNSNUMBER</td>
<td>=</td>
<td>msdyn_deliveryaddressdunsnumber</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSESLATITUDE</td>
<td>=</td>
<td>msdyn_deliveryaddresslatitude</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSLOCATIONID</td>
<td>=</td>
<td>msdyn_deliveryaddresslocationid</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSLONGITUDE</td>
<td>=</td>
<td>msdyn_deliveryaddresslongitude</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSNAME</td>
<td>=</td>
<td>msdyn_deliveryaddressname</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSPOSTBOX</td>
<td>=</td>
<td>msdyn_deliveryaddresspostbox</td>
<td></td>
</tr>
<tr>
<td>DELIVERYBUILDINGCOMPLEMENT</td>
<td>=</td>
<td>msdyn_deliverybuildingcomplement</td>
<td></td>
</tr>
<tr>
<td>EXTERNALITEMNUMBER</td>
<td>=</td>
<td>msdyn_externalitemnumber</td>
<td></td>
</tr>
<tr>
<td>FIXEDPRICECHARGES</td>
<td>=</td>
<td>msdyn_fixedpricecharges</td>
<td></td>
</tr>
<tr>
<td>FORMATTEDADDRESSED</td>
<td>=</td>
<td>msdyn_formattedaddressed</td>
<td></td>
</tr>
<tr>
<td>LINEDESCRIPTION</td>
<td>=</td>
<td>msdyn_linedescription</td>
<td></td>
</tr>
<tr>
<td>LINEDISCOUNTAMOUNT</td>
<td>=</td>
<td>msdyn_linediscountamount</td>
<td></td>
</tr>
<tr>
<td>LINEDISCOUNTPERCENTAGE</td>
<td>=</td>
<td>msdyn_linediscountpercentage</td>
<td></td>
</tr>
<tr>
<td>MULTILINEIDISCOUNTAMOUNT</td>
<td>=</td>
<td>msdyn_multilineidiscountamount</td>
<td></td>
</tr>
<tr>
<td>MULTILINEIDISCOUNTPERCENTAGE</td>
<td>=</td>
<td>msdyn_multilineidiscountpercentage</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>REQUESTEDRECEIPTDATE</td>
<td>=</td>
<td>msdyn_requestedreceiptdate</td>
<td></td>
</tr>
<tr>
<td>REQUESTEDSHIPPINGDATE</td>
<td>=</td>
<td>requestdeliveryby</td>
<td></td>
</tr>
<tr>
<td>SALESORDERLINestatus</td>
<td>&gt;&gt;</td>
<td>msdyn_linestatus</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>none: 192350000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>backorder: 192350001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>delivered: 192350002</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>invoiced: 192350003</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>canceled: 192350004</td>
<td></td>
</tr>
<tr>
<td>SALESORDERPROMISINGMETHOD</td>
<td>&lt;</td>
<td>msdyn_salesorderpromisingmethod</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>none: 192350000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>salesLeadTime: 192350001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>atp: 192350002</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>atpPlusIssueMargin: 192350003</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ctp: 192350004</td>
<td></td>
</tr>
<tr>
<td>SALESPRICEQUANTITY</td>
<td>=</td>
<td>msdyn_salespricequantity</td>
<td></td>
</tr>
<tr>
<td>SHIPPINGSITEID</td>
<td>=</td>
<td>msdyn_shippingsite.msdyn_siteid</td>
<td></td>
</tr>
<tr>
<td>SHIPPINGWAREHOUSEID</td>
<td>=</td>
<td>msdyn_shippingwarehouse. msdyn_warehouseidentifier</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>none: &gt;&gt;</td>
<td>ispriceoverridden</td>
</tr>
<tr>
<td>TOTALCHARGESAMOUNT</td>
<td>&gt;</td>
<td>msdyn_totalchargesamount</td>
<td></td>
</tr>
<tr>
<td>DELIVERYMODECODE</td>
<td>=</td>
<td>msdyn_deliverymode.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>DELIVERYTERMSID</td>
<td>=</td>
<td>msdyn_deliveryterms.msdyn_termsscode</td>
<td></td>
</tr>
</tbody>
</table>

**CDS sales quotation header (quotes)**

This template synchronizes data between Finance and Operations apps and Dataverse.

Reversed source filter: msdyn_ordertype eq 192350000 and statecode eq 0

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SALESQUOTATIONNUMBER</td>
<td>=</td>
<td>msdyn_quotenumber</td>
<td></td>
</tr>
<tr>
<td>REQUESTINGCUSTOMERACCOUNTNUMBER</td>
<td>=</td>
<td>customerid.Account(accountnumber).Contact(msdyn_contactpersonid)</td>
<td></td>
</tr>
<tr>
<td>Finance and Operations Field</td>
<td>Map Type</td>
<td>Customer Engagement Column</td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------</td>
<td>-----------------------------</td>
<td></td>
</tr>
<tr>
<td>CurrencyCode</td>
<td>=</td>
<td>transactioncurrencyid.isocurrencycode</td>
<td></td>
</tr>
<tr>
<td>CustomersReference</td>
<td>=</td>
<td>msdyn_customersreference</td>
<td></td>
</tr>
<tr>
<td>DeliveryAddressCity</td>
<td>=</td>
<td>shipto_city</td>
<td></td>
</tr>
<tr>
<td>DeliveryAddressCountryRegionIsoCode</td>
<td>&lt;</td>
<td>shipto_country</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>us: united states</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>de: germany</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>fr: france</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>gb: united kingdom</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>jp: japan</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>in: india</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>es: spain</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>nz: new zealand</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>au: australia</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>co: colombia</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>za: south africa</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>be: belgium</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ca: canada</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>at: austria</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>tr: turkey</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>cn: china</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>dk: denmark</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>se: sweden</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>no: norway</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>fl: finland</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>eg: egypt</td>
<td></td>
</tr>
<tr>
<td>DeliveryAddressStreetNumber</td>
<td>=</td>
<td>shipto_line2</td>
<td></td>
</tr>
<tr>
<td>DeliveryAddressZipCode</td>
<td>=</td>
<td>shipto_postalcode</td>
<td></td>
</tr>
<tr>
<td>DeliveryAddressStreet</td>
<td>=</td>
<td>shipto_line1</td>
<td></td>
</tr>
<tr>
<td>DeliveryAddressStateId</td>
<td>=</td>
<td>shipto_stateorprovince</td>
<td></td>
</tr>
<tr>
<td>SalesQuotationName</td>
<td>=</td>
<td>name</td>
<td></td>
</tr>
<tr>
<td>InvoiceAddressCity</td>
<td>&gt;</td>
<td>billto_city</td>
<td></td>
</tr>
<tr>
<td>InvoiceAddressStreet</td>
<td>&gt;</td>
<td>billto_line1</td>
<td></td>
</tr>
<tr>
<td>InvoiceAddressStreetNumber</td>
<td>&gt;</td>
<td>billto_line2</td>
<td></td>
</tr>
<tr>
<td>InvoiceAddressCountryRegionIsoCode</td>
<td>&gt;</td>
<td>billto_country</td>
<td></td>
</tr>
<tr>
<td>InvoiceAddressStateId</td>
<td>&gt;</td>
<td>billto_stateorprovince</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>INVOICEADDRESSZIPCODE</td>
<td>&gt;</td>
<td>billto_postalcode</td>
<td></td>
</tr>
<tr>
<td>QUOTATIONTOTALAMOUNT</td>
<td>&gt;</td>
<td>totalamount</td>
<td></td>
</tr>
<tr>
<td>TOTALDISCOUNTAMOUNT</td>
<td>&gt;</td>
<td>discountamount</td>
<td></td>
</tr>
<tr>
<td>QUOTATIONTOTALTAXAMOUNT</td>
<td>&gt;</td>
<td>totaltax</td>
<td></td>
</tr>
<tr>
<td>QUOTATIONTOTALCHARGEAMOUNT</td>
<td>&gt;</td>
<td>freightamount</td>
<td></td>
</tr>
<tr>
<td>AREPRICESINCLUDINGSALESTAX</td>
<td>&lt;</td>
<td>msdyn_arepricesincludingsalesax</td>
<td>False, True</td>
</tr>
<tr>
<td>CONTACTPERSONID</td>
<td>=</td>
<td>msdyn_associatedcontact.msdy_contactforpartynumber</td>
<td></td>
</tr>
<tr>
<td>CUSTOMERREQUISITIONNUMBER</td>
<td>=</td>
<td>msdyn_customerrequisitionnumber</td>
<td></td>
</tr>
<tr>
<td>DEFAULTSHIPPINGSITEID</td>
<td>=</td>
<td>msdyn_defaultshippingsite.msdyn_siteid</td>
<td></td>
</tr>
<tr>
<td>DEFAULTSHIPPINGWAREHOUSEID</td>
<td>=</td>
<td>msdyn_defaultshippingwareHOUSE.msdyn_warehousedentifier</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSCOUNTYID</td>
<td>=</td>
<td>msdyn_deliveryaddresscounttyid</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSDESCRIPTION</td>
<td>=</td>
<td>msdyn_deliveryaddressdescription</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSDISTRICTNAME</td>
<td>=</td>
<td>msdyn_deliveryaddressdistrictname</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSDUNSNUMBER</td>
<td>=</td>
<td>msdyn_deliveryaddressdunsnumber</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSLATITUDE</td>
<td>=</td>
<td>msdyn_deliveryaddresslatitude</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSLOCATIONID</td>
<td>=</td>
<td>msdyn_deliveryaddresslocationid</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSLONGITUDE</td>
<td>=</td>
<td>msdyn_deliveryaddresslongitude</td>
<td></td>
</tr>
<tr>
<td><strong>F I E L D</strong></td>
<td><strong>M A P T I P E</strong></td>
<td><strong>C O L U M N</strong></td>
<td><strong>D E F A U L T V A L U E</strong></td>
</tr>
<tr>
<td>------------------------------</td>
<td>------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>DELIVERYADDRESSNAME</td>
<td>=</td>
<td>msdyn_deliveryaddressname</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSPOSTBOX</td>
<td>=</td>
<td>msdyn_deliveryaddresspostbox</td>
<td></td>
</tr>
<tr>
<td>DELIVERYBUILDINGCOMPLIMENT</td>
<td>=</td>
<td>msdyn_deliverybuildingcompliment</td>
<td></td>
</tr>
<tr>
<td>FORMATTEDDELIVERYADDRESS</td>
<td>&gt;</td>
<td>msdyn_formatteddeliveryaddress</td>
<td></td>
</tr>
<tr>
<td>FORMATTEDINVOICEADDRESS</td>
<td>&gt;</td>
<td>msdyn_formattedinvoiceaddress</td>
<td></td>
</tr>
<tr>
<td>GENERATEDSALESDORDERNUMBER</td>
<td>=</td>
<td>msdyn_generatedsalessordernumber</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSCOUNTRYID</td>
<td>&gt;</td>
<td>msdyn_invoiceaddresscountryid</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSCOUNTYID</td>
<td>&gt;</td>
<td>msdyn_invoiceaddresscountyid</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSDISTRICTNAME</td>
<td>&gt;</td>
<td>msdyn_invoiceaddressdistrictname</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSLATITUDE</td>
<td>&gt;</td>
<td>msdyn_invoiceaddresslatitude</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSLONGITUDE</td>
<td>&gt;</td>
<td>msdyn_invoiceaddresslongitude</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSPOSTBOX</td>
<td>&gt;</td>
<td>msdyn_invoiceaddresspostbox</td>
<td></td>
</tr>
<tr>
<td>INVOICEBUILDINGCOMPLIMENT</td>
<td>&gt;</td>
<td>msdyn_invoicebuildingcompliment</td>
<td></td>
</tr>
<tr>
<td>INVOICECUSTOMERACCOUNTNUMBER</td>
<td>=</td>
<td>msdyn_invoicecustomerid.Account(accountnumber).Contact(msdyn_contactpersonid)</td>
<td></td>
</tr>
<tr>
<td>ISDELIVERYADDRESSORDERSPECIFIC</td>
<td>&gt; &lt;</td>
<td>msdyn_isdeliveryaddressorderspecific</td>
<td></td>
</tr>
<tr>
<td>ISDELIVERYADDRESSPRIVATE</td>
<td>&gt; &lt;</td>
<td>msdyn_isdeliveryaddressprivate</td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>MAP TYPE</td>
<td>COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------</td>
<td>-----------------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>ISINVOICEADDRESSPRIVATE</td>
<td>&gt;&gt;</td>
<td>msdyn_isinvoiceaddressprivate</td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>False</td>
<td>msdyn_isinvoiceaddressprivate</td>
<td></td>
</tr>
<tr>
<td>yes</td>
<td>True</td>
<td>msdyn_isinvoiceaddressprivate</td>
<td></td>
</tr>
<tr>
<td>LANGUAGEID</td>
<td>&gt; &lt;</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>ar</td>
<td>192350000</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>ar-ae</td>
<td>192350001</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>cs</td>
<td>192350002</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>da</td>
<td>192350003</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>de</td>
<td>192350004</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>de-at</td>
<td>192350005</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>de-ch</td>
<td>192350006</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>en-au</td>
<td>192350007</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>en-gb</td>
<td>192350009</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>en-ie</td>
<td>192350010</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>en-in</td>
<td>192350011</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>en-ca</td>
<td>192350008</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>en-my</td>
<td>192350012</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>en-nz</td>
<td>192350013</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>en-sg</td>
<td>192350014</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>en-us</td>
<td>192350015</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>en-za</td>
<td>192350016</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>es</td>
<td>192350017</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>es-mx</td>
<td>192350018</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>et</td>
<td>192350019</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>fi</td>
<td>192350020</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>fr</td>
<td>192350021</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>fr-be</td>
<td>192350022</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>fr-ca</td>
<td>192350023</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>fr-ch</td>
<td>192350024</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>hu</td>
<td>192350025</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>is</td>
<td>192350026</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>it</td>
<td>192350027</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>it-ch</td>
<td>192350028</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>ja</td>
<td>192350029</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>lt</td>
<td>192350030</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>lv</td>
<td>192350031</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>nb-no</td>
<td>192350032</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>nl</td>
<td>192350033</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>nl-be</td>
<td>192350034</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>pl</td>
<td>192350035</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>pt-br</td>
<td>192350036</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>ru</td>
<td>192350037</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>sv</td>
<td>192350038</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>th</td>
<td>192350039</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>tr</td>
<td>192350040</td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td>zh-hans</td>
<td>192350041</td>
<td>msdyn_language</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FIELD</th>
<th>=</th>
<th>COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAYMENTTERMSNAME</td>
<td>=</td>
<td>msdyn_paymentterms.msdy_paymentterms</td>
<td></td>
</tr>
<tr>
<td>PRICECUSTOMERGROUPCODE</td>
<td>=</td>
<td>msdyn_pricecustomergroupcode</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT FIELD</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>---------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>RECEIPTDATERequested</td>
<td>=</td>
<td>msdyn_requestedreceiptdate</td>
<td></td>
</tr>
<tr>
<td>REQUESTEDSHIPPINGDATE</td>
<td>=</td>
<td>requestdeliveryby</td>
<td></td>
</tr>
<tr>
<td>SALESORDERPROMISINGMETHOD</td>
<td>&gt; &lt;</td>
<td>msdyn_salesorderpromisingmethod</td>
<td></td>
</tr>
<tr>
<td></td>
<td>none : 192350000</td>
<td>salesLeadTime : 192350001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>atp : 192350002</td>
<td>atpPlusIssueMargin : 192350003</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ctp : 192350004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SALESQUOTATIONCONFIRMATIONDATE</td>
<td>=</td>
<td>msdyn_salesquotationconfirmationdate</td>
<td></td>
</tr>
<tr>
<td>SALESQUOTATIONEXPIRYDATE</td>
<td>=</td>
<td>msdyn_salesquotationexpirydate</td>
<td></td>
</tr>
<tr>
<td>SALESQUOTATIONFOLLOWUPDATE</td>
<td>=</td>
<td>msdyn_salesquotationfollowupdate</td>
<td></td>
</tr>
<tr>
<td>SALESQUOTATIONSTATUS</td>
<td>&gt; &lt;</td>
<td>msdyn_salesquotationstatus</td>
<td></td>
</tr>
<tr>
<td></td>
<td>created : 192350000</td>
<td>sent : 192350001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>confirmed : 192350002</td>
<td>lost : 192350003</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cancelled : 192350004</td>
<td>reset : 192350005</td>
<td></td>
</tr>
<tr>
<td></td>
<td>modified : 192350006</td>
<td>submitted : 192350007</td>
<td></td>
</tr>
<tr>
<td></td>
<td>approved : 192350008</td>
<td>revised : 192350009</td>
<td></td>
</tr>
<tr>
<td>TOTALDISCOUNTPERCENTAGE</td>
<td>=</td>
<td>msdyn_totaldiscountpercentage</td>
<td></td>
</tr>
<tr>
<td>URL</td>
<td>=</td>
<td>msdyn_url</td>
<td></td>
</tr>
<tr>
<td>EMAIL</td>
<td>=</td>
<td>emailaddress</td>
<td></td>
</tr>
<tr>
<td>none</td>
<td>&gt;&gt;</td>
<td>msdyn_ordertype</td>
<td>1923500000</td>
</tr>
<tr>
<td>CURRENCYCODE</td>
<td>&gt;</td>
<td>msdyn_isocurrencycode</td>
<td></td>
</tr>
</tbody>
</table>
| OPERATINGUNITPARTYNUMBER     | =        | msdyn_operatingunit.msdy
n_partynumber |               |

**CDS sales quotation lines (quotedetails)**

This template synchronizes data between Finance and Operations apps and Dataverse.
<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLOWEDOVERDELIVERYPERCENTAGE</td>
<td>=</td>
<td>msdyn_allowedoverdeliverypercentage</td>
<td></td>
</tr>
<tr>
<td>ALLOWEDUNDERDELIVERYPERCENTAGE</td>
<td>=</td>
<td>msdyn_allowedunderdeliverypercentage</td>
<td></td>
</tr>
<tr>
<td>SALESQUOTATIONNUMBER</td>
<td>=</td>
<td>quoteid.msdyn_quotenumber</td>
<td></td>
</tr>
<tr>
<td>LINECREATIONSEQUENCENUMBER</td>
<td>=</td>
<td>sequencenumber</td>
<td></td>
</tr>
<tr>
<td>CURRENCYCODE</td>
<td>&gt;</td>
<td>transactioncurrencyid.isocurrency</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSCITY</td>
<td>=</td>
<td>shipto_city</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSCOUNTR YREGIONISOCODE</td>
<td>=&lt;</td>
<td>shipto_country</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSCOUNTYID</td>
<td>=</td>
<td>msdyn_deliveryaddresscoun tyid</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSDESCRIPTION</td>
<td>=</td>
<td>msdyn_deliveryaddressdescription</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSDISTRICTNAME</td>
<td>=</td>
<td>msdyn_deliveryaddressdistrictname</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSDUNSNUMBER</td>
<td>=</td>
<td>msdyn_deliveryaddressdunsnumber</td>
<td></td>
</tr>
<tr>
<td>Finance and Operations Field</td>
<td>Map Type</td>
<td>Customer Engagement Field</td>
<td>Default Value</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------</td>
<td>---------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>DeliveryAddressLatitude</td>
<td>=</td>
<td>msdyn_deliveryaddresslatitude</td>
<td></td>
</tr>
<tr>
<td>DeliveryAddressLocationID</td>
<td>=</td>
<td>msdyn_deliveryaddresslocationid</td>
<td></td>
</tr>
<tr>
<td>DeliveryAddressLongitude</td>
<td>=</td>
<td>msdyn_deliveryaddresslongitude</td>
<td></td>
</tr>
<tr>
<td>DeliveryAddressName</td>
<td>=</td>
<td>msdyn_deliveryaddressname</td>
<td></td>
</tr>
<tr>
<td>DeliveryAddressPostbox</td>
<td>=</td>
<td>msdyn_deliveryaddresspostbox</td>
<td></td>
</tr>
<tr>
<td>DeliveryAddressStateID</td>
<td>=</td>
<td>shipto_stateorprovince</td>
<td></td>
</tr>
<tr>
<td>DeliveryAddressStreet</td>
<td>=</td>
<td>shipto_line1</td>
<td></td>
</tr>
<tr>
<td>DeliveryAddressStreetNumber</td>
<td>=</td>
<td>shipto_line2</td>
<td></td>
</tr>
<tr>
<td>DeliveryAddressZipCode</td>
<td>=</td>
<td>shipto_postalcode</td>
<td></td>
</tr>
<tr>
<td>DeliveryBuildingCompliment</td>
<td>=</td>
<td>msdyn_deliverybuildingcompliment</td>
<td></td>
</tr>
<tr>
<td>ExternalItemNumber</td>
<td>=</td>
<td>msdyn_externalitemnumber</td>
<td></td>
</tr>
<tr>
<td>FixedPriceCharges</td>
<td>=</td>
<td>msdyn_fixedpricecharges</td>
<td></td>
</tr>
<tr>
<td>FormattedDeliveryAddress</td>
<td>=</td>
<td>msdyn_formatteddeliveryaddress</td>
<td></td>
</tr>
<tr>
<td>IsDeliveryAddressPrivate</td>
<td>&gt; &lt;</td>
<td>msdyn_isdeliveryaddressprivate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td>yes : True</td>
<td></td>
</tr>
<tr>
<td>IsDeliveryAddressOrderSpecifc</td>
<td>&gt; &lt;</td>
<td>msdyn_isdeliveryaddressspecifc</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td>yes : True</td>
<td></td>
</tr>
<tr>
<td>LineAmount</td>
<td>&gt;</td>
<td>baseamount</td>
<td></td>
</tr>
<tr>
<td>LineAmount</td>
<td>&gt;</td>
<td>extendedamount</td>
<td></td>
</tr>
<tr>
<td>LineDescription</td>
<td>=</td>
<td>msdyn_linedescription</td>
<td></td>
</tr>
<tr>
<td>LineDiscountAmount</td>
<td>=</td>
<td>msdyn_linediscountamount</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>LINEDISCOUNTPERCENTAGE</td>
<td>=</td>
<td>msdyn_linediscountpercentage</td>
<td></td>
</tr>
<tr>
<td>MULTILINEDISCOUNTAMOUNT</td>
<td>=</td>
<td>msdyn_multilinediscountamount</td>
<td></td>
</tr>
<tr>
<td>MULTILINEDISCOUNTPERCENTAGE</td>
<td>=</td>
<td>msdyn_multilinediscountpercentage</td>
<td></td>
</tr>
<tr>
<td>PRODUCTNAME</td>
<td>=</td>
<td>description</td>
<td></td>
</tr>
<tr>
<td>PRODUCTNUMBER</td>
<td>=</td>
<td>productid.msdyn_productnumber</td>
<td></td>
</tr>
<tr>
<td>REQUESTEDRECEIPTDATE</td>
<td>=</td>
<td>msdyn_requestedreceiptdate</td>
<td></td>
</tr>
<tr>
<td>REQUESTEDSALESQUANTITY</td>
<td>=</td>
<td>quantity</td>
<td></td>
</tr>
<tr>
<td>REQUESTEDSHIPPINGDATE</td>
<td>=</td>
<td>requestdeliveryby</td>
<td></td>
</tr>
<tr>
<td>SALESPRICE</td>
<td>=</td>
<td>priceperunit</td>
<td></td>
</tr>
<tr>
<td>SALESPRICEQUANTITY</td>
<td>=</td>
<td>msdyn_salespricequantity</td>
<td></td>
</tr>
<tr>
<td>SALESQUOTATIONPROMISINGMETHOD</td>
<td>&gt; &lt;</td>
<td>msdyn_salesquotationpromisingmethod</td>
<td></td>
</tr>
<tr>
<td></td>
<td>none : 192350000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>salesLeadTime : 192350001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>atp : 192350002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>atpPlusIssueMargin : 192350003</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ctp : 192350004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SALESQUOTATIONSTATUS</td>
<td>&gt; &lt;</td>
<td>msdyn_salesquotationstatus</td>
<td></td>
</tr>
<tr>
<td></td>
<td>created : 192350000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sent : 192350001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>confirmed : 192350002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lost : 192350003</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>cancelled : 192350004</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset : 192350005</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>modified : 192350006</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>submitted : 192350007</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>approved : 192350008</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>revised : 192350009</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SALEUNITSYMBOL</td>
<td>=</td>
<td>uomid.msdyn_symbol</td>
<td></td>
</tr>
<tr>
<td>SHIPPINGSITEID</td>
<td>=</td>
<td>msdyn_shippingsite.msdyn_siteid</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>SHIPPINGWAREHOUSEID</td>
<td>=</td>
<td>msdyn_shippingwarehouse. msdyn_warehouseidentifier</td>
<td></td>
</tr>
<tr>
<td>TOTALCHARGESAMOUNT</td>
<td>&gt;</td>
<td>msdyn_totalchargesamount</td>
<td></td>
</tr>
<tr>
<td>TOTALDISCOUNTAMOUNT</td>
<td>=</td>
<td>manualdiscountamount</td>
<td></td>
</tr>
<tr>
<td>TOTALTAXAMOUNT</td>
<td>&gt;</td>
<td>tax</td>
<td></td>
</tr>
<tr>
<td>SALESPRODUCTCATEGORY</td>
<td>&gt;</td>
<td>msdyn_salesproductcategory.name</td>
<td></td>
</tr>
<tr>
<td>HIERARCHYNAME</td>
<td>=</td>
<td>msdyn_salesproductcategory.name</td>
<td></td>
</tr>
<tr>
<td>none</td>
<td>&gt;&gt;</td>
<td>ispriceoverridden</td>
<td>true</td>
</tr>
</tbody>
</table>

**Chart of accounts (msdyn_chartofaccountses)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
<tr>
<td>MAINACCOUNTMASK</td>
<td>=</td>
<td>msdyn_mainaccountmask</td>
<td></td>
</tr>
<tr>
<td>CHARTOFACCOUNTS</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
</tbody>
</table>

**Colors (msdyn_productcolors)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLORID</td>
<td>&gt;</td>
<td>msdyn_productcolorname</td>
<td></td>
</tr>
</tbody>
</table>

**Compensation job function (cdm_jobfunctions)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOBFUNCTIONID</td>
<td>=</td>
<td>cdm_name</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>cdm_description</td>
<td></td>
</tr>
</tbody>
</table>

**Compensation job type (cdm_jobtypes)**

This template synchronizes data between Finance and Operations apps and Dataverse.
<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOBTYPEID</td>
<td>=</td>
<td>cdm_name</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>cdm_description</td>
<td></td>
</tr>
<tr>
<td>EXEMPTSTATUS</td>
<td>&gt; &lt;</td>
<td>cdm_exemptstatus</td>
<td></td>
</tr>
<tr>
<td></td>
<td>exempt  : 754400000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>nonexempt : 754400001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>doesnotapply : 754400002</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Complimentary closings (msdyn_complimentaryclosings)
This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOSINGPHRASE</td>
<td>=</td>
<td>msdyn_closingphrase</td>
<td></td>
</tr>
</tbody>
</table>

Configurations (msdyn_productconfigurations)
This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIGURATIONID</td>
<td>&gt;</td>
<td>msdyn_productconfiguration</td>
<td></td>
</tr>
</tbody>
</table>

Contact person titles (msdyn_salescontactpersontitles)
This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOBTITLE</td>
<td>=</td>
<td>msdyn_jobtitle</td>
<td></td>
</tr>
<tr>
<td>JOBTITLEALIAS</td>
<td>=</td>
<td>msdyn_jobtitlealias</td>
<td></td>
</tr>
</tbody>
</table>

Contacts V2 (msdyn_contactforparties)
This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTACTPERSONID</td>
<td>=</td>
<td>msdyn_contactforpartynumber</td>
<td></td>
</tr>
<tr>
<td>CONTACTPERSONPARTYNUMBER</td>
<td>=</td>
<td>msdyn_contactpartyid.msdy n_partynumber</td>
<td></td>
</tr>
<tr>
<td>ASSOCIATEDPARTYNUMBER</td>
<td>=</td>
<td>msdyn_associatedpartyid.m sdyn_partynumber</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>EMPLOYMENTPROFESSION</td>
<td>=</td>
<td>msdyn_employmentprofession</td>
<td></td>
</tr>
<tr>
<td>EMPLOYMENTDEPARTMENT</td>
<td>=</td>
<td>msdyn_employmentdepartment</td>
<td></td>
</tr>
<tr>
<td>PRIMARYSALUTATIONPHRASE</td>
<td>=</td>
<td>msdyn_primarysalutationphrase.msdyn_salutationphrase</td>
<td></td>
</tr>
<tr>
<td>ASSISTANTNAME</td>
<td>=</td>
<td>msdyn_assistantname</td>
<td></td>
</tr>
<tr>
<td>ASSISTANTPHONENUMBER</td>
<td>=</td>
<td>msdyn_assistantphonenumber</td>
<td></td>
</tr>
<tr>
<td>AVAILABLEFROMTIME</td>
<td>=</td>
<td>msdyn_availablefromtime</td>
<td></td>
</tr>
<tr>
<td>AVAILABLETOTIME</td>
<td>=</td>
<td>msdyn_availabletotime</td>
<td></td>
</tr>
<tr>
<td>CONTACTACTIVITYSENSITIVITYLEVEL</td>
<td>&lt;</td>
<td>msdyn_contactactivitysensitivitylevel</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>normal: 192350000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>personal: 192350001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>private: 192350002</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>confidential: 192350003</td>
<td></td>
</tr>
<tr>
<td>CONTACTPERSONRESPONSIBLEPERSONNELNUMBER</td>
<td>=</td>
<td>msdyn_employeeresponsible.cdm_workernumber</td>
<td></td>
</tr>
<tr>
<td>DECISIONMAKINGROLECODE</td>
<td>=</td>
<td>msdyn_decisionmakingrole.msdyn_rolename</td>
<td></td>
</tr>
<tr>
<td>EMPLOYMENTCOMPUTERNETWORKNAME</td>
<td>=</td>
<td>msdyn_employmentcomputernetworkname</td>
<td></td>
</tr>
<tr>
<td>EMPLOYMENTJOBFUNCTIONNAME</td>
<td>=</td>
<td>msdyn_employmentjobfunctionname</td>
<td></td>
</tr>
<tr>
<td>EMPLOYMENTJOBTITLE</td>
<td>=</td>
<td>msdyn_jobtitle.msdyn_jobtitlealias</td>
<td></td>
</tr>
<tr>
<td>EMPLOYMENTOFFICELOCATION</td>
<td>=</td>
<td>msdyn_employmentofficelocation</td>
<td></td>
</tr>
<tr>
<td>GOVERNMENTIDENTIFICATIONNUMBER</td>
<td>=</td>
<td>msdyn_governmentidentificationnumber</td>
<td></td>
</tr>
<tr>
<td>HASREQUESTEDINTERNETACCESS</td>
<td>&gt; &lt;</td>
<td>msdyn_hasrequestedinternetaccess</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>no: False</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>yes: True</td>
<td></td>
</tr>
<tr>
<td>IDENTITCARDNUMBER</td>
<td>=</td>
<td>msdyn_identitycardnumber</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Map Type</td>
<td>Column</td>
<td>Default Value</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>----------</td>
<td>---------------------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>ISRECEIVINGDIRECTMAIL</td>
<td>&gt; &lt;</td>
<td>msdyn_isreceivingdirectmail</td>
<td></td>
</tr>
<tr>
<td>ISVIP</td>
<td>&gt; &lt;</td>
<td>msdyn_isvip</td>
<td></td>
</tr>
<tr>
<td>LOYALTYLEVELPHRASE</td>
<td>=</td>
<td>msdyn_loyaltylevelphrase.msdyn_levelphrase</td>
<td></td>
</tr>
<tr>
<td>MANAGERCONTACTPERSONID</td>
<td>=</td>
<td>msdyn_parentcontactforpartyid.msdyn_contactforpartyid.msdyn_contactforpartyid.number</td>
<td></td>
</tr>
<tr>
<td>MICROSOFTOUTLOOKCATEGORIES</td>
<td>=</td>
<td>msdyn_microsoftoutlookcategories</td>
<td></td>
</tr>
<tr>
<td>MILEAGEDISTANCE</td>
<td>=</td>
<td>msdyn_mileagedistance</td>
<td></td>
</tr>
<tr>
<td>NOTES</td>
<td>=</td>
<td>msdyn_notes</td>
<td></td>
</tr>
<tr>
<td>ORGANIZATIONIDENTIFICATIONNUMBER</td>
<td>=</td>
<td>msdyn_organizationidentificationnumber</td>
<td></td>
</tr>
<tr>
<td>PERSONALCHARACTERTYPECODE</td>
<td>=</td>
<td>msdyn_personalcharactertypecode.msdyn_typename</td>
<td></td>
</tr>
<tr>
<td>PRIMARYCOMPLIMENTARYCLOSINGPHRASE</td>
<td>=</td>
<td>msdyn_primarycomplimentaryclosingphrase.msdyn_closingphrase</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT FIELD</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------</td>
<td>---------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>CONTACTINFORMATIONLANGUAGEID</td>
<td>&gt;=</td>
<td>msdyn_nativelanguage</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ar</td>
<td></td>
<td>192350000</td>
</tr>
<tr>
<td></td>
<td>ar-ae</td>
<td></td>
<td>192350001</td>
</tr>
<tr>
<td></td>
<td>cs</td>
<td></td>
<td>192350002</td>
</tr>
<tr>
<td></td>
<td>da</td>
<td></td>
<td>192350003</td>
</tr>
<tr>
<td></td>
<td>de</td>
<td></td>
<td>192350004</td>
</tr>
<tr>
<td></td>
<td>de-at</td>
<td></td>
<td>192350005</td>
</tr>
<tr>
<td></td>
<td>de-ch</td>
<td></td>
<td>192350006</td>
</tr>
<tr>
<td></td>
<td>en-au</td>
<td></td>
<td>192350007</td>
</tr>
<tr>
<td></td>
<td>en-gb</td>
<td></td>
<td>192350009</td>
</tr>
<tr>
<td></td>
<td>en-ie</td>
<td></td>
<td>192350010</td>
</tr>
<tr>
<td></td>
<td>en-in</td>
<td></td>
<td>192350011</td>
</tr>
<tr>
<td></td>
<td>en-ca</td>
<td></td>
<td>192350008</td>
</tr>
<tr>
<td></td>
<td>en-my</td>
<td></td>
<td>192350012</td>
</tr>
<tr>
<td></td>
<td>en-nz</td>
<td></td>
<td>192350013</td>
</tr>
<tr>
<td></td>
<td>en-sg</td>
<td></td>
<td>192350014</td>
</tr>
<tr>
<td></td>
<td>en-us</td>
<td></td>
<td>192350015</td>
</tr>
<tr>
<td></td>
<td>en-za</td>
<td></td>
<td>192350016</td>
</tr>
<tr>
<td></td>
<td>es</td>
<td></td>
<td>192350017</td>
</tr>
<tr>
<td></td>
<td>es-mx</td>
<td></td>
<td>192350018</td>
</tr>
<tr>
<td></td>
<td>et</td>
<td></td>
<td>192350019</td>
</tr>
<tr>
<td></td>
<td>fi</td>
<td></td>
<td>192350020</td>
</tr>
<tr>
<td></td>
<td>fr</td>
<td></td>
<td>192350021</td>
</tr>
<tr>
<td></td>
<td>fr-be</td>
<td></td>
<td>192350022</td>
</tr>
<tr>
<td></td>
<td>fr-ca</td>
<td></td>
<td>192350023</td>
</tr>
<tr>
<td></td>
<td>fr-ch</td>
<td></td>
<td>192350024</td>
</tr>
<tr>
<td></td>
<td>hu</td>
<td></td>
<td>192350025</td>
</tr>
<tr>
<td></td>
<td>is</td>
<td></td>
<td>192350026</td>
</tr>
<tr>
<td></td>
<td>it</td>
<td></td>
<td>192350027</td>
</tr>
<tr>
<td></td>
<td>it-ch</td>
<td></td>
<td>192350028</td>
</tr>
<tr>
<td></td>
<td>ja</td>
<td></td>
<td>192350029</td>
</tr>
<tr>
<td></td>
<td>it</td>
<td></td>
<td>192350030</td>
</tr>
<tr>
<td></td>
<td>lv</td>
<td></td>
<td>192350031</td>
</tr>
<tr>
<td></td>
<td>nb-no</td>
<td></td>
<td>192350032</td>
</tr>
<tr>
<td></td>
<td>nl</td>
<td></td>
<td>192350033</td>
</tr>
<tr>
<td></td>
<td>nl-be</td>
<td></td>
<td>192350034</td>
</tr>
<tr>
<td></td>
<td>pl</td>
<td></td>
<td>192350035</td>
</tr>
<tr>
<td></td>
<td>pt-br</td>
<td></td>
<td>192350036</td>
</tr>
<tr>
<td></td>
<td>ru</td>
<td></td>
<td>192350037</td>
</tr>
<tr>
<td></td>
<td>sv</td>
<td></td>
<td>192350038</td>
</tr>
<tr>
<td></td>
<td>th</td>
<td></td>
<td>192350039</td>
</tr>
<tr>
<td></td>
<td>tr</td>
<td></td>
<td>192350040</td>
</tr>
<tr>
<td></td>
<td>zh-hans</td>
<td></td>
<td>192350041</td>
</tr>
<tr>
<td>SPOUSENAME</td>
<td>=</td>
<td>msdyn_spousename</td>
<td></td>
</tr>
</tbody>
</table>

**Currencies (transactioncurrencies)**

This template synchronizes data between Finance and Operations apps and Dataverse.

Source filter: ((CURRENCYCODE != "999"))
Customer Attachments (annotations)

This template synchronizes data between Finance and Operations apps and Dataverse.

Source filter: ((Typeld eq "Note") || (Typeld eq "URL"))

Reversed source filter: (objecttypecode eq 'account' or objecttypecode eq 'contact') and msdyn_relatedentityid ne null

Customer groups (msdyn_customergroups)

This template synchronizes data between Finance and Operations apps and Dataverse.
<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAYMENTTERMID</td>
<td>=</td>
<td>msdyn_paymenttermid.msdyntermname</td>
<td></td>
</tr>
<tr>
<td>CLEARINGPERIOPDpaymentTERMNAME</td>
<td>=</td>
<td>msdyn_clearingperiodpaymenttermname.msdyntermname</td>
<td></td>
</tr>
</tbody>
</table>

**Customer hierarchies (msdyn_customerhierarchies)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOMERHIERARCHYID</td>
<td>=</td>
<td>msdyn_customerhierarchynumber</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>ORGANIZATIONPARTYNUMBER</td>
<td>=</td>
<td>msdyn_organizationpartynumber.msdyn_partynumber</td>
<td></td>
</tr>
<tr>
<td>PURPOSE</td>
<td>&gt; &lt;</td>
<td>msdyn_purpose</td>
<td></td>
</tr>
</tbody>
</table>

**Customer hierarchy nodes (msdyn_customerhierarchynodes)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOMERHIERARCHYID</td>
<td>=</td>
<td>msdyn_customerhierarchynumber.msdyn_customerhierarchynumber</td>
<td></td>
</tr>
<tr>
<td>NODEPARTYNUMBER</td>
<td>=</td>
<td>msdyn_nodepartynumber.msdyn_partynumber</td>
<td></td>
</tr>
<tr>
<td>NODETYPE</td>
<td>&gt; &lt;</td>
<td>msdyn_nodetype</td>
<td></td>
</tr>
<tr>
<td>ROLE</td>
<td>&gt; &lt;</td>
<td>msdyn_role</td>
<td></td>
</tr>
</tbody>
</table>

**Customer payment method (msdyn_customerpaymentmethods)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
</tbody>
</table>

This template synchronizes data between Finance and Operations apps and Dataverse.
<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCOUNTTYPE</td>
<td>&gt; &lt;</td>
<td></td>
<td>msdyn_accounttype</td>
</tr>
<tr>
<td>DISCOUNTGRACEPERIODDAYS</td>
<td>=</td>
<td></td>
<td>msdyn_discountgraceperioddays</td>
</tr>
<tr>
<td>BRIDGINGPOSTINGENABLED</td>
<td>&gt; &lt;</td>
<td></td>
<td>msdyn_bridgingpostingenabled</td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td>yes : True</td>
<td></td>
</tr>
<tr>
<td>ISSEPA</td>
<td>&gt; &lt;</td>
<td></td>
<td>msdyn_issepa</td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td>yes : True</td>
<td></td>
</tr>
<tr>
<td>LASTFILENUMBER</td>
<td>=</td>
<td></td>
<td>msdyn_lastfilenumber</td>
</tr>
<tr>
<td>LASTFILENUMBERTODAY</td>
<td>=</td>
<td></td>
<td>msdyn_lastfilenumber today</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td></td>
<td>msdyn_description</td>
</tr>
<tr>
<td>PAYMENTTYPE</td>
<td>&gt; &lt;</td>
<td></td>
<td>msdyn_paymenttype</td>
</tr>
<tr>
<td>CREATEANDDRAWBILL OF EXCHANGE CHANGEDURINGINVOICEPOSTING</td>
<td>&gt; &lt;</td>
<td></td>
<td>msdyn_invoiceupdate</td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td>yes : True</td>
<td></td>
</tr>
<tr>
<td>PAYMENTSTATUS</td>
<td>&gt; &lt;</td>
<td></td>
<td>msdyn_paymentstatus</td>
</tr>
<tr>
<td></td>
<td>none : 806380000</td>
<td>sent : 806380001</td>
<td>received : 806380002</td>
</tr>
<tr>
<td>SUMBYPERIOD</td>
<td>&gt; &lt;</td>
<td></td>
<td>msdyn_sumbyperiod</td>
</tr>
<tr>
<td></td>
<td>invoice : 806380000</td>
<td>week : 806380002</td>
<td>total : 806380003</td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>-----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>ENABLEPOSTDATEDCHECK CLEARINGPOSTING</td>
<td>&gt; &lt;</td>
<td>msdyn_enablepostdatescheckclearingposting</td>
<td></td>
</tr>
<tr>
<td>BILLOFEXCHANGEDRAFTYPE</td>
<td>&gt; &lt;</td>
<td>msdyn_billofexchangedraftype</td>
<td></td>
</tr>
<tr>
<td>DIRECTDEBIT</td>
<td>&gt; &lt;</td>
<td>msdyn_directdebit</td>
<td></td>
</tr>
</tbody>
</table>

**Customers V3 (accounts)**

This template synchronizes data between Finance and Operations apps and Dataverse.

Source filter: (PartyType == "Organization")

Reversed source filter: customertypecode eq 3

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOMERACCOUNT</td>
<td>=</td>
<td>accountnumber</td>
<td></td>
</tr>
<tr>
<td>CREDITLIMIT</td>
<td>=</td>
<td>creditlimit</td>
<td></td>
</tr>
<tr>
<td>SALESCURRENCYCODE</td>
<td>=</td>
<td>transactioncurrencyid.isocurrencycodename</td>
<td></td>
</tr>
<tr>
<td>SALESMEMO</td>
<td>=</td>
<td>description</td>
<td></td>
</tr>
<tr>
<td>CREDITLIMITISMANDATORY</td>
<td>&gt; &lt;</td>
<td>msdyn_creditlimitismandatory</td>
<td></td>
</tr>
<tr>
<td>CREDITRATING</td>
<td>=</td>
<td>msdyn_creditsrating</td>
<td></td>
</tr>
<tr>
<td>CUSTOMERGROUPID</td>
<td>=</td>
<td>msdyn_customergroupid.msdyn_groupid</td>
<td></td>
</tr>
<tr>
<td>IDENTIFICATIONNUMBER</td>
<td>=</td>
<td>msdyn_identificationnumber</td>
<td></td>
</tr>
<tr>
<td>INVOICEACCOUNT</td>
<td>=</td>
<td>msdyn_billingaccount.accountnumber</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------</td>
<td>-----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>ISONETIMECUSTOMER</td>
<td>&lt;=</td>
<td>msdyn_onetimecustomer</td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>yes</td>
<td>True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ONHOLDSTATUS</td>
<td>&lt;=</td>
<td>msdyn_onholdstatus</td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>806380000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>invoice</td>
<td>806380001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>all</td>
<td>806380002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>payment</td>
<td>806380003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>requisition</td>
<td>806380004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>never</td>
<td>806380005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PARTYCOUNTRY</td>
<td>=</td>
<td>msdyn_partycountry</td>
<td></td>
</tr>
<tr>
<td>PARTystate</td>
<td>=</td>
<td>msdyn_partystateprovince</td>
<td></td>
</tr>
<tr>
<td>PAYMENTDAY</td>
<td>=</td>
<td>msdyn_paymentday.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>PAYMENTMETHOD</td>
<td>=</td>
<td>msdyn_customerpaymentmethod.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>PAYMENTSCHEDULE</td>
<td>=</td>
<td>msdyn_paymentschedule.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>PAYMENTTERMS</td>
<td>=</td>
<td>msdyn_paymentterm.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>PAYMENTTERMSBASEDAYS</td>
<td>=</td>
<td>msdyn_paymenttermsbasedays</td>
<td></td>
</tr>
<tr>
<td>TAXEXEMPTNUMBER</td>
<td>=</td>
<td>msdyn_taxexemptnumber</td>
<td></td>
</tr>
<tr>
<td>VENDORACCOUNT</td>
<td>=</td>
<td>msdyn_vendor.msdyn_vendoraccountnumber</td>
<td></td>
</tr>
<tr>
<td>none</td>
<td>=&gt;</td>
<td>customertypecode</td>
<td>3</td>
</tr>
<tr>
<td>PARTYTYPE</td>
<td>&lt;=</td>
<td>none</td>
<td>Organization</td>
</tr>
<tr>
<td>PARTYNNUMBER</td>
<td>=</td>
<td>msdyn_partyid.msdyn_partynumber</td>
<td></td>
</tr>
<tr>
<td>CONTACTPERSONID</td>
<td>=</td>
<td>msdyn_primarycontact.msdyn_contactforpartynumber</td>
<td></td>
</tr>
<tr>
<td>SALESTAXGROUP</td>
<td>=</td>
<td>msdyn_salestaxgroup.msdyn_name</td>
<td></td>
</tr>
</tbody>
</table>

**Customers V3 (contacts)**

This template synchronizes data between Finance and Operations apps and Dataverse.
Source filter: ($(PartyType == "Person")

Reversed source filter: msdyn_sellable eq true and msdyn_contactpersonid ne null

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>&gt; &gt;</td>
<td>msdyn_sellable</td>
<td>True</td>
</tr>
<tr>
<td>PARTYTYPE</td>
<td>&lt;=</td>
<td>none</td>
<td>Person</td>
</tr>
<tr>
<td>PARTYNUMBER</td>
<td>=</td>
<td>msdyn_partyid.msdyn_partyid.msdyn_partynumber</td>
<td></td>
</tr>
<tr>
<td>CUSTOMERACCOUNT</td>
<td>=</td>
<td>msdyn_contactpersonid</td>
<td></td>
</tr>
<tr>
<td>CUSTOMERGROUPID</td>
<td>=</td>
<td>msdyn_customergroupid.msdyn_groupid</td>
<td></td>
</tr>
<tr>
<td>IDENTIFICATIONNUMBER</td>
<td>=</td>
<td>msdyn_identificationnumber</td>
<td></td>
</tr>
<tr>
<td>PARTYCOUNTRY</td>
<td>=</td>
<td>msdyn_partycountry</td>
<td></td>
</tr>
<tr>
<td>PARTystate</td>
<td>=</td>
<td>msdyn_partystateprovince</td>
<td></td>
</tr>
<tr>
<td>SALESCURRENCYCODE</td>
<td>=</td>
<td>transactioncurrencyid.isocurrencycode</td>
<td></td>
</tr>
<tr>
<td>SALESMEMO</td>
<td>=</td>
<td>description</td>
<td></td>
</tr>
<tr>
<td>PAYMENTDAY</td>
<td>=</td>
<td>msdyn_paymentday.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>PAYMENTSCHEDULE</td>
<td>=</td>
<td>msdyn_paymentschedule.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>PAYMENTMETHOD</td>
<td>=</td>
<td>msdyn_customerpaymentmethod.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>SALESTAXGROUP</td>
<td>=</td>
<td>msdyn_salestaxgroup.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>PAYMENTTERMS</td>
<td>=</td>
<td>msdyn_paymentterms.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>CONTACTPERSONID</td>
<td>=</td>
<td>msdyn_primarycontact.msdyn_contactforpartynumber</td>
<td></td>
</tr>
</tbody>
</table>

**Decision making roles (msdyn_decisionmakingroles)**

This template synchronizes data between Finance and Operations apps and Dataverse.
### FINANCE AND OPERATIONS

<table>
<thead>
<tr>
<th>FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROLEDESCRIPTION</td>
<td>=</td>
<td>msdyn_roledescription</td>
<td></td>
</tr>
<tr>
<td>ROLENAME</td>
<td>=</td>
<td>msdyn_rolename</td>
<td></td>
</tr>
</tbody>
</table>

### Default order settings (msdyn_productdefaultordersettings)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVENTWAREHOUSEID</td>
<td>=</td>
<td>msdyn_inventorywarehouse.msdyn_warehouseidentifier</td>
<td></td>
</tr>
<tr>
<td>INVENTORYSITEID</td>
<td>=</td>
<td>msdyn_inventorysite.msdyn_siteid</td>
<td></td>
</tr>
<tr>
<td>INVENTORYATPDELAYEDDEMANDOFFSETDAYS</td>
<td>=</td>
<td>msdyn_inventoryatpdelayed_demandoffsetdays</td>
<td></td>
</tr>
<tr>
<td>INVENTORYATPDELAYEDSUPPLYOFFSETDAYS</td>
<td>=</td>
<td>msdyn_inventoryatpdelayed_supplyoffsetdays</td>
<td></td>
</tr>
<tr>
<td>ITEMNUMBER</td>
<td>=</td>
<td>msdyn_itemnumber.msdyn_itemnumber</td>
<td></td>
</tr>
<tr>
<td>INVENTORYATPBACKWARDDEMANDTIMEFENCEDAYS</td>
<td>=</td>
<td>msdyn_inventoryatpbackward_demandtimefencedays</td>
<td></td>
</tr>
<tr>
<td>INVENTORYATPBACKWARDSUPPLYTIMEFENCEDAYS</td>
<td>=</td>
<td>msdyn_inventoryatpbackward_supplytimefencedays</td>
<td></td>
</tr>
<tr>
<td>INVENTORYATPTIMEFENCEDAYS</td>
<td>=</td>
<td>msdyn_inventoryatptimefencedays</td>
<td></td>
</tr>
<tr>
<td>MAXIMUMINVENTORYORDERQUANTITY</td>
<td>=</td>
<td>msdyn_maximuminventoryorderquantity</td>
<td></td>
</tr>
<tr>
<td>MAXIMUMPROCUREMENTORDERQUANTITY</td>
<td>=</td>
<td>msdyn_maximumprocurementorderquantity</td>
<td></td>
</tr>
<tr>
<td>MAXIMUMSALESORDERQUANTITY</td>
<td>=</td>
<td>msdyn_maximumsalesorderquantity</td>
<td></td>
</tr>
<tr>
<td>MINIMUMINVENTORYORDERQUANTITY</td>
<td>=</td>
<td>msdyn_minimuminventoryorderquantity</td>
<td></td>
</tr>
<tr>
<td>MINIMUMPROCUREMENTORDERQUANTITY</td>
<td>=</td>
<td>msdyn_minimumprocurementorderquantity</td>
<td></td>
</tr>
<tr>
<td>MINIMUMSALESORDERQUANTITY</td>
<td>=</td>
<td>msdyn_minimumsalesorderquantity</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Map Type</td>
<td>Customer Engagement Column</td>
<td>Default Value</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>standardinventoryorderquantity</td>
<td>=</td>
<td>msdyn_standardinventoryorderquantity</td>
<td></td>
</tr>
<tr>
<td>standardprocurementorderquantity</td>
<td>=</td>
<td>msdyn_standardprocurementorderquantity</td>
<td></td>
</tr>
<tr>
<td>standardsalesorderid</td>
<td>=</td>
<td>msdyn_standardsalesorderid</td>
<td></td>
</tr>
<tr>
<td>inventoryleadtimedays</td>
<td>=</td>
<td>msdyn_inventoryleadtimedays</td>
<td></td>
</tr>
<tr>
<td>inventoryquantitymultiples</td>
<td>=</td>
<td>msdyn_inventoryquantitymultiples</td>
<td></td>
</tr>
<tr>
<td>procurementquantitymultiples</td>
<td>=</td>
<td>msdynProcurementQuantityMultiples</td>
<td></td>
</tr>
<tr>
<td>salesquantitymultiples</td>
<td>=</td>
<td>msdyn_salesquantitymultiples</td>
<td></td>
</tr>
<tr>
<td>procurementsiteid</td>
<td>=</td>
<td>msdyn_procurementsite.msdyn_siteid</td>
<td></td>
</tr>
<tr>
<td>procurementleadtimedays</td>
<td>=</td>
<td>msdyn_procurementleadtimedays</td>
<td></td>
</tr>
<tr>
<td>salesiteid</td>
<td>=</td>
<td>msdyn_salesite.msdyn_siteid</td>
<td></td>
</tr>
<tr>
<td>salesatpdelayeddemandoffsetdays</td>
<td>=</td>
<td>msdyn_salesatpdelayeddemandoffsetdays</td>
<td></td>
</tr>
<tr>
<td>salesatpdelayedsupplyoffsetdays</td>
<td>=</td>
<td>msdyn_salesatpdelayedsupplyoffsetdays</td>
<td></td>
</tr>
<tr>
<td>salesatpbackwarddemandtimefencedays</td>
<td>=</td>
<td>msdyn_salesatpbackwarddemandtimefencedays</td>
<td></td>
</tr>
<tr>
<td>salesatpbackwardsupplytimefencedays</td>
<td>=</td>
<td>msdyn_salesatpbackwardsupplytimefencedays</td>
<td></td>
</tr>
<tr>
<td>salesatptimefencedays</td>
<td>=</td>
<td>msdyn_salesatptimefencedays</td>
<td></td>
</tr>
<tr>
<td>salesleadtimedays</td>
<td>=</td>
<td>msdyn_salesleadtimedays</td>
<td></td>
</tr>
<tr>
<td>procurementwarehouseid</td>
<td>=</td>
<td>msdyn_procurementwarehouse.msdyn_warehouseidentifier</td>
<td></td>
</tr>
<tr>
<td>saleswarehouseid</td>
<td>=</td>
<td>msdyn_saleswarehouse.msdyn_warehouseidentifier</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT FIELD</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>AREINVENTORYORDERDEFAULTSOVERRIDEN</td>
<td>&gt; &lt;</td>
<td>msdyn_areinventoryorderdeaultsoverridden</td>
<td></td>
</tr>
<tr>
<td>INVENTORYORDERPROMISINGMETHOD</td>
<td>&gt; &lt;</td>
<td>msdyn_inventoryorderpromisingmethod</td>
<td></td>
</tr>
<tr>
<td>ISINVENTORYATPINCLUDINGPLANNEDORDERS</td>
<td>&gt; &lt;</td>
<td>msdyn_isinventoryatpincludingplannedorders</td>
<td></td>
</tr>
<tr>
<td>ISINVENTORYUSINGWORKINGDAYS</td>
<td>&gt; &lt;</td>
<td>msdyn_isinventoryusingworkingdays</td>
<td></td>
</tr>
<tr>
<td>ISINVENTORIESITEMANDATORY</td>
<td>&gt; &lt;</td>
<td>msdyn_isinventoriesitemandatory</td>
<td></td>
</tr>
<tr>
<td>ISINVENTORYPROCESSINGSTOPPED</td>
<td>&gt; &lt;</td>
<td>msdyn_isinventoryprocessingstopped</td>
<td></td>
</tr>
<tr>
<td>ISPROCUREMENTUSINGWORKINGDAYS</td>
<td>&gt; &lt;</td>
<td>msdyn_isprocurementusingworkingdays</td>
<td></td>
</tr>
<tr>
<td>ISPROCUREMENTSITEMANDATORY</td>
<td>&gt; &lt;</td>
<td>msdyn_isprocurementsitemandatory</td>
<td></td>
</tr>
<tr>
<td>ISPROCUREMENTPROCESSINGSTOPPED</td>
<td>&gt; &lt;</td>
<td>msdyn_isprocurementprocessingstopped</td>
<td></td>
</tr>
<tr>
<td>ARESALESORDERPROMISINGDEFAULTSOVERRIDEN</td>
<td>&gt; &lt;</td>
<td>msdyn_areasalesorderdefaultsoverridden</td>
<td></td>
</tr>
<tr>
<td>SALESORDERPROMISINGMETHOD</td>
<td>&gt; &lt;</td>
<td>msdyn_salesorderpromisingmethod</td>
<td></td>
</tr>
</tbody>
</table>
### Employment job functions (msdyn_employmentjobfunctions)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FunctionDescription</td>
<td>=</td>
<td>msdyn_functiondescription</td>
<td></td>
</tr>
<tr>
<td>FunctionName</td>
<td>=</td>
<td>msdyn_functionname</td>
<td></td>
</tr>
</tbody>
</table>

### Employment per company (cdm_employments)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmploymentEnddate</td>
<td>=</td>
<td>cdm_employmentenddate</td>
<td></td>
</tr>
<tr>
<td>PersonnelNumber</td>
<td>=</td>
<td>cdm_workerid.cdm_workerid</td>
<td></td>
</tr>
<tr>
<td>EmploymentStartDate</td>
<td>=</td>
<td>cdm_employmentstartdate</td>
<td></td>
</tr>
</tbody>
</table>
### WORKERTYPE

<table>
<thead>
<tr>
<th>Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>employee</td>
<td>&gt;&gt;</td>
<td>cdm_workertype</td>
<td></td>
</tr>
<tr>
<td>contractor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>both</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Ethnic origins (cdm_ethnicorigins)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETHNICORIGINID</td>
<td>=</td>
<td>cdm_name</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>cdm_description</td>
<td></td>
</tr>
</tbody>
</table>

#### Exchange rate currency pair (msdyn_currencyexchangeratepairs)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCHANGERATEDISPLAYFACTOR</td>
<td>&gt; &lt;</td>
<td>msdyn_displayfactor</td>
<td></td>
</tr>
</tbody>
</table>

#### Exchange rate type (msdyn_exchangeratetypes)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
</tbody>
</table>

#### Financial dimension format (msdyn_financialdimensionformats)

This template synchronizes data between Finance and Operations apps and Dataverse.
### Financial dimensions (msdyn_dimensionattributes)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIMENSIONNAME</td>
<td>=</td>
<td>msdyn_dimensionname</td>
<td></td>
</tr>
<tr>
<td>COPYVALUESONCREATE</td>
<td>&lt;</td>
<td>msdyn_copyvaluesoncreate</td>
<td></td>
</tr>
<tr>
<td>no:</td>
<td>False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>yes:</td>
<td>True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REPORTCOLUMNNAME</td>
<td>=</td>
<td>msdyn_reportcolumnname</td>
<td></td>
</tr>
<tr>
<td>GIVEDERIVEDDIMENSIONS PRECEDENCE</td>
<td>&lt;</td>
<td>msdyn_givederiveddimensionsprecedence</td>
<td></td>
</tr>
<tr>
<td>no:</td>
<td>False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>yes:</td>
<td>True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UseValuesFrom</td>
<td>=</td>
<td>msdyn_usevaluesfrom</td>
<td></td>
</tr>
<tr>
<td>DimensionValueMask</td>
<td>=</td>
<td>msdyn_dimensionvaluemask</td>
<td></td>
</tr>
</tbody>
</table>

### Fiscal calendar integration entity (msdyn_fiscalcalendars)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALENDARID</td>
<td>=</td>
<td>msdyn_calendar</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
</tbody>
</table>
### Fiscal calendar period (msdyn_fiscalcalendarperiods)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMENTS</td>
<td>=</td>
<td>msdyn_comments</td>
<td></td>
</tr>
<tr>
<td>ENDDATE</td>
<td>=</td>
<td>msdyn_enddate</td>
<td></td>
</tr>
<tr>
<td>MONTH</td>
<td>&gt; &lt;</td>
<td>msdyn_month</td>
<td></td>
</tr>
<tr>
<td></td>
<td>month1</td>
<td>192350000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>month2</td>
<td>192350001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>month3</td>
<td>192350002</td>
<td></td>
</tr>
<tr>
<td></td>
<td>month4</td>
<td>192350003</td>
<td></td>
</tr>
<tr>
<td></td>
<td>month5</td>
<td>192350004</td>
<td></td>
</tr>
<tr>
<td></td>
<td>month6</td>
<td>192350005</td>
<td></td>
</tr>
<tr>
<td></td>
<td>month7</td>
<td>192350006</td>
<td></td>
</tr>
<tr>
<td></td>
<td>month8</td>
<td>192350007</td>
<td></td>
</tr>
<tr>
<td></td>
<td>month9</td>
<td>192350008</td>
<td></td>
</tr>
<tr>
<td></td>
<td>month10</td>
<td>192350009</td>
<td></td>
</tr>
<tr>
<td></td>
<td>month11</td>
<td>192350010</td>
<td></td>
</tr>
<tr>
<td></td>
<td>month12</td>
<td>192350011</td>
<td></td>
</tr>
<tr>
<td>CALENDAR</td>
<td>=</td>
<td>msdyn_fiscalcalendar.msdyn_calendar</td>
<td></td>
</tr>
<tr>
<td>QUARTER</td>
<td>&gt; &lt;</td>
<td>msdyn_quarter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>q1</td>
<td>192350000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>q2</td>
<td>192350001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>q3</td>
<td>192350002</td>
<td></td>
</tr>
<tr>
<td></td>
<td>q4</td>
<td>192350003</td>
<td></td>
</tr>
<tr>
<td>SHORTNAME</td>
<td>=</td>
<td>msdyn_shortname</td>
<td></td>
</tr>
<tr>
<td>STARTDATE</td>
<td>=</td>
<td>msdyn_startdate</td>
<td></td>
</tr>
<tr>
<td>TYPE</td>
<td>&gt; &lt;</td>
<td>msdyn_fiscalperiodtype</td>
<td></td>
</tr>
<tr>
<td></td>
<td>opening</td>
<td>192350000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>operating</td>
<td>192350001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>closing</td>
<td>192350002</td>
<td></td>
</tr>
<tr>
<td>PERIODNAME</td>
<td>=</td>
<td>msdyn_periodname</td>
<td></td>
</tr>
<tr>
<td>FISCALYEAR</td>
<td>=</td>
<td>msdyn_fiscalcalendaryear.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>CALENDAR</td>
<td>=</td>
<td>msdyn_fiscalcalendaryear.msdyn_fiscalcalendarname</td>
<td></td>
</tr>
</tbody>
</table>

### Fiscal calendar year integration entity (msdyn_fiscalcalendaryears)

This template synchronizes data between Finance and Operations apps and Dataverse.
<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FISCALCALENDAR_CALENDAR_ARID</td>
<td>=</td>
<td>msdyn_fiscalcalendarnname</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>STARTDATE</td>
<td>=</td>
<td>msdyn_startdate</td>
<td></td>
</tr>
<tr>
<td>ENDDATE</td>
<td>=</td>
<td>msdyn_enddate</td>
<td></td>
</tr>
<tr>
<td>FISCALCALENDAR_CALENDAR_ARID</td>
<td>=</td>
<td>msdyn_calendar.msdyn_calendar</td>
<td></td>
</tr>
</tbody>
</table>

**Inventory aisle (msdyn_warehouseaisles)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>AISLEID</td>
<td>=</td>
<td>msdyn_aisleid</td>
<td></td>
</tr>
<tr>
<td>AISLENUMBER</td>
<td>=</td>
<td>msdyn_aislenumber</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEID</td>
<td>=</td>
<td>msdyn_warehouse.msdyn_warehouseid</td>
<td></td>
</tr>
<tr>
<td>AISLENAME</td>
<td>=</td>
<td>msdyn_aislename</td>
<td></td>
</tr>
<tr>
<td>MANUALSTARTINGSORTORDERCODE</td>
<td>=</td>
<td>msdyn_manualstartingsortordercode</td>
<td></td>
</tr>
<tr>
<td>ISSORTORDERCODEASSIGNEDEDDECENDING</td>
<td>&gt; &lt;</td>
<td>msdyn_issortordercodeassigneddescending</td>
<td></td>
</tr>
</tbody>
</table>

**Item sales tax group (msdyn_taxitemgroups)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAXITEMGROUP</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
</tbody>
</table>

**Jobs (cdm_jobs)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOBID</td>
<td>=</td>
<td>cdm_name</td>
<td></td>
</tr>
<tr>
<td><strong>FINANCE AND OPERATIONS FIELD</strong></td>
<td><strong>MAP TYPE</strong></td>
<td><strong>CUSTOMER ENGAGEMENT COLUMN</strong></td>
<td><strong>DEFAULT VALUE</strong></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------</td>
<td>-------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>MAXIMUMNUMBEROFPOSITIONS</td>
<td>=</td>
<td>cdm_maximumnumberofpositions</td>
<td></td>
</tr>
<tr>
<td>ALLOWUNLIMITEDPOSITIONS</td>
<td>&gt;</td>
<td>cdm_allowunlimitedpositions</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>cdm_description</td>
<td></td>
</tr>
<tr>
<td>JOBDESCRIPTION</td>
<td>=</td>
<td>cdm_jobdescription</td>
<td></td>
</tr>
<tr>
<td>JOBTYPEID</td>
<td>=</td>
<td>cdm_jobtypeid.cdm_name</td>
<td></td>
</tr>
<tr>
<td>FUNCTIONID</td>
<td>=</td>
<td>cdm_jobfunctionid.cdm_name</td>
<td></td>
</tr>
<tr>
<td>EFFECTIVE</td>
<td>=</td>
<td>cdm_validfrom</td>
<td></td>
</tr>
<tr>
<td>EXPIRATION</td>
<td>=</td>
<td>cdm_validto</td>
<td></td>
</tr>
<tr>
<td>FULLTIMEEQUIVALENT</td>
<td>=</td>
<td>cdm_defaultfulltimeequivale nt</td>
<td></td>
</tr>
</tbody>
</table>

**Language codes (cdm_languages)**
This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th><strong>FINANCE AND OPERATIONS FIELD</strong></th>
<th><strong>MAP TYPE</strong></th>
<th><strong>CUSTOMER ENGAGEMENT COLUMN</strong></th>
<th><strong>DEFAULT VALUE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>LANGUAGECODEID</td>
<td>=</td>
<td>cdm_name</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>cdm_description</td>
<td></td>
</tr>
</tbody>
</table>

**Ledger (msdyn_ledgers)**
This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th><strong>FINANCE AND OPERATIONS FIELD</strong></th>
<th><strong>MAP TYPE</strong></th>
<th><strong>CUSTOMER ENGAGEMENT COLUMN</strong></th>
<th><strong>DEFAULT VALUE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>LEGALENTITYID</td>
<td>&gt;</td>
<td>msdyn_company.cdm_companycode</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>&gt;</td>
<td>msdyn_description</td>
<td></td>
</tr>
<tr>
<td>ACCOUNTINGCURRENCY</td>
<td>&gt;</td>
<td>msdyn_accountingcurrency.currencycod e</td>
<td></td>
</tr>
<tr>
<td>ISBUDGETCONTROLENABLED</td>
<td>&gt;&gt;</td>
<td>msdyn_isbudgetcontroled</td>
<td></td>
</tr>
</tbody>
</table>
### Legal entities (cdm_companies)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>&gt;</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>LEGALENTITYID</td>
<td>&gt;</td>
<td>cdm_companycode</td>
<td></td>
</tr>
</tbody>
</table>

### Legal entities (msdyn_internalorganizations)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAMEALIAS</td>
<td>&gt;</td>
<td>msdyn_namealias</td>
<td></td>
</tr>
<tr>
<td>LANGUAGEID</td>
<td>&gt;</td>
<td>msdyn_languageid</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>&gt;</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>PARTYNUMBER</td>
<td>&gt;</td>
<td>msdyn_partynumber</td>
<td></td>
</tr>
<tr>
<td>none</td>
<td>&gt;&gt;</td>
<td>msdyn_type</td>
<td>806380000</td>
</tr>
<tr>
<td>LEGALENTITYID</td>
<td>&gt;</td>
<td>msdyn_companycode</td>
<td></td>
</tr>
</tbody>
</table>

### Loyalty card (msdyn_loyaltycards)

This template synchronizes data between Finance and Operations apps and Dataverse.
<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARDNUMBER</td>
<td>=</td>
<td>msdyn_cardnumber</td>
<td></td>
</tr>
<tr>
<td>CARDTENDERTYPE</td>
<td>&gt; &lt;</td>
<td>msdyn_carDtendertype</td>
<td></td>
</tr>
<tr>
<td></td>
<td>asCardTender : 806380000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>asContactTender : 806380001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>noTender : 806380002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>blocked : 806380003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PARTYNUMBER</td>
<td>=</td>
<td>msdyn_partynumber</td>
<td></td>
</tr>
<tr>
<td>REPLACEMENTCARDNUMBER</td>
<td>&gt;</td>
<td>msdyn_replacementcardnumber</td>
<td></td>
</tr>
<tr>
<td>OPERATINGUNITNUMBER</td>
<td>=</td>
<td>msdyn_operatingunitnumber</td>
<td></td>
</tr>
<tr>
<td>LOYALTYENROLLMENTDATE</td>
<td>=</td>
<td>msdyn_enrollmentdate</td>
<td></td>
</tr>
<tr>
<td>LOYALTYENROLLMENTDATELOCAL</td>
<td>=</td>
<td>msdyn_effectivedate</td>
<td></td>
</tr>
</tbody>
</table>

**Loyalty levels (msdyn_loyaltylevels)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEVELPHRASE</td>
<td>=</td>
<td>msdyn_levelphrase</td>
<td></td>
</tr>
<tr>
<td>LEVELDESCRIPTION</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
</tbody>
</table>

**Loyalty reward points (msdyn_loyaltyrewardpoints)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXPIRATIONTIMEUNIT</td>
<td>&gt; &lt;</td>
<td>msdyn_expirationtimeunit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>day : 806380000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>month : 806380001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>year : 806380002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPIRATIONTIMEVALUE</td>
<td>=</td>
<td>msdyn_expirationtimevalue</td>
<td></td>
</tr>
<tr>
<td>REDEEMABLE</td>
<td>&gt; &lt;</td>
<td>msdyn_redeemable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REDEEMRANKING</td>
<td>=</td>
<td>msdyn_redeemranking</td>
<td></td>
</tr>
<tr>
<td><strong>FINANCE AND OPERATIONS FIELD</strong></td>
<td><strong>MAP TYPE</strong></td>
<td><strong>CUSTOMER ENGAGEMENT COLUMN</strong></td>
<td><strong>DEFAULT VALUE</strong></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------</td>
<td>---------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>REWARDPOINTCURRENCY</td>
<td>=</td>
<td>msdyn_rewardpointcurrenc y.isocurrencycode</td>
<td></td>
</tr>
<tr>
<td>REWARDPOINTID</td>
<td>=</td>
<td>msdyn_rewardpointid</td>
<td></td>
</tr>
<tr>
<td>REWARDPOINTTYPE</td>
<td>&lt;</td>
<td>quantity : 192350000</td>
<td>192350001</td>
</tr>
<tr>
<td>MAXIMUMLOYALTYREWARDPOINTS</td>
<td>=</td>
<td>msdyn_maximumloyaltyrewardpoints</td>
<td></td>
</tr>
<tr>
<td>VESTINGTIMEUNIT</td>
<td>&lt;</td>
<td>day : 806380000</td>
<td>806380001</td>
</tr>
<tr>
<td>VESTINGTIMEVALUE</td>
<td>=</td>
<td>msdyn_vestingtimevalue</td>
<td></td>
</tr>
</tbody>
</table>

**Main account (msdyn_mainaccounts)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th><strong>FINANCE AND OPERATIONS FIELD</strong></th>
<th><strong>MAP TYPE</strong></th>
<th><strong>CUSTOMER ENGAGEMENT COLUMN</strong></th>
<th><strong>DEFAULT VALUE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>MAINACCOUNTID</td>
<td>=</td>
<td>msdyn_accountnumber</td>
<td></td>
</tr>
<tr>
<td>CHARTOFACCOUNTS</td>
<td>=</td>
<td>msdyn_chartofaccounts.ms dyn_name</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>BALANCECONTROL</td>
<td>&lt;</td>
<td>none : 192350000</td>
<td>192350000</td>
</tr>
<tr>
<td>EXCHANGEADJUSTMENTRATETYPE</td>
<td>=</td>
<td>msdyn_exchangeadjustmentratetype.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>CLOSING</td>
<td>&lt;</td>
<td>none : 192350000</td>
<td>192350000</td>
</tr>
<tr>
<td>REPORTINGEXCHANGEADJUSTMENTRATETYPE</td>
<td>=</td>
<td>msdyn_reportingexchangeadjustmentratetype.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>MAP TYPE</td>
<td>COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>----------</td>
<td>-------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>DEBITCREDITREQUIREMENT</td>
<td>=&gt;</td>
<td>msdyn_debitcreditrequirement</td>
<td></td>
</tr>
<tr>
<td></td>
<td>none</td>
<td>192350000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>debit</td>
<td>192350001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>credit</td>
<td>192350002</td>
<td></td>
</tr>
<tr>
<td>FINANCIALREPORTINGEXCHANGERATETYPE</td>
<td>=</td>
<td>msdyn_financialreportingexchangeratetype.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>FOREIGNCURRENCYREVALUATION</td>
<td>=&gt;</td>
<td>msdyn_foreigncurrencyrevaluation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>MAINACCOUNTCATEGORY</td>
<td>=</td>
<td>msdyn_mainaccountcategory.name</td>
<td></td>
</tr>
<tr>
<td>MANDATORYPAYMENTREFERENCE</td>
<td>=&gt;</td>
<td>msdyn_mandatorypaymentreference</td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>MONETARY</td>
<td>=&gt;</td>
<td>msdyn_monetary</td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>OFFSETACCOUNTDISPLAYVALUE</td>
<td>=</td>
<td>msdyn_offsetaccount</td>
<td></td>
</tr>
<tr>
<td>POSTINGTYPE</td>
<td>=&gt;</td>
<td>msdyn_postingtype</td>
<td></td>
</tr>
<tr>
<td></td>
<td>none</td>
<td>192350000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>exchRateGain</td>
<td>192350001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>exchRateLoss</td>
<td>192350002</td>
<td></td>
</tr>
<tr>
<td></td>
<td>interCompany</td>
<td>192350003</td>
<td></td>
</tr>
<tr>
<td></td>
<td>tax</td>
<td>192350004</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VATRoundOff</td>
<td>192350005</td>
<td></td>
</tr>
<tr>
<td></td>
<td>allocation</td>
<td>192350006</td>
<td></td>
</tr>
<tr>
<td></td>
<td>investmentDuty</td>
<td>192350007</td>
<td></td>
</tr>
<tr>
<td></td>
<td>liquidity</td>
<td>192350008</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mSTDiffSecond</td>
<td>192350009</td>
<td></td>
</tr>
<tr>
<td></td>
<td>errorAccount</td>
<td>192350010</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mSTDiff</td>
<td>192350011</td>
<td></td>
</tr>
<tr>
<td></td>
<td>yearResult</td>
<td>192350012</td>
<td></td>
</tr>
<tr>
<td></td>
<td>closing</td>
<td>192350013</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ledgerJournal</td>
<td>192350014</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cashDiscount</td>
<td>192350015</td>
<td></td>
</tr>
<tr>
<td></td>
<td>consolidateDiffBalance</td>
<td>192350016</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------</td>
<td>-----------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>transferOpeningClosing</td>
<td>192350016</td>
<td>transferOpeningClosing</td>
<td></td>
</tr>
<tr>
<td>bank</td>
<td>192350018</td>
<td>bank</td>
<td></td>
</tr>
<tr>
<td>conversionProfit</td>
<td>192350019</td>
<td>conversionProfit</td>
<td></td>
</tr>
<tr>
<td>conversionLoss</td>
<td>192350020</td>
<td>conversionLoss</td>
<td></td>
</tr>
<tr>
<td>taxwithhold</td>
<td>192350021</td>
<td>taxwithhold</td>
<td></td>
</tr>
<tr>
<td>consolidateDiffProfitLoss</td>
<td>192350022</td>
<td>consolidateDiffProfitLoss</td>
<td></td>
</tr>
<tr>
<td>indirectEstimatedAbsorptionOffset</td>
<td>192350023</td>
<td>indirectEstimatedAbsorptionOffset</td>
<td></td>
</tr>
<tr>
<td>indirectAbsorption</td>
<td>192350024</td>
<td>indirectAbsorption</td>
<td></td>
</tr>
<tr>
<td>indirectAbsorptionOffset</td>
<td>192350025</td>
<td>indirectAbsorptionOffset</td>
<td></td>
</tr>
<tr>
<td>freeTextInvoice</td>
<td>192350026</td>
<td>freeTextInvoice</td>
<td></td>
</tr>
<tr>
<td>conversionReportingLoss</td>
<td>192350027</td>
<td>conversionReportingLoss</td>
<td></td>
</tr>
<tr>
<td>conversionReportingProfit</td>
<td>192350029</td>
<td>conversionReportingProfit</td>
<td></td>
</tr>
<tr>
<td>custBalance</td>
<td>192350030</td>
<td>custBalance</td>
<td></td>
</tr>
<tr>
<td>custRevenue</td>
<td>192350031</td>
<td>custRevenue</td>
<td></td>
</tr>
<tr>
<td>custInterest</td>
<td>192350032</td>
<td>custInterest</td>
<td></td>
</tr>
<tr>
<td>custCashDisc</td>
<td>192350033</td>
<td>custCashDisc</td>
<td></td>
</tr>
<tr>
<td>custCollectionLetterFee</td>
<td>192350034</td>
<td>custCollectionLetterFee</td>
<td></td>
</tr>
<tr>
<td>custInterestFee</td>
<td>192350035</td>
<td>custInterestFee</td>
<td></td>
</tr>
<tr>
<td>custInvoiceDisc</td>
<td>192350036</td>
<td>custInvoiceDisc</td>
<td></td>
</tr>
<tr>
<td>custPayment</td>
<td>192350037</td>
<td>custPayment</td>
<td></td>
</tr>
<tr>
<td>custReimbursement</td>
<td>192350038</td>
<td>custReimbursement</td>
<td></td>
</tr>
<tr>
<td>custSettlement</td>
<td>192350039</td>
<td>custSettlement</td>
<td></td>
</tr>
<tr>
<td>vendBalance</td>
<td>192350040</td>
<td>vendBalance</td>
<td></td>
</tr>
<tr>
<td>vendPurchLedger</td>
<td>192350041</td>
<td>vendPurchLedger</td>
<td></td>
</tr>
<tr>
<td>vendOffsetAccount</td>
<td>192350042</td>
<td>vendOffsetAccount</td>
<td></td>
</tr>
<tr>
<td>vendInterest</td>
<td>192350043</td>
<td>vendInterest</td>
<td></td>
</tr>
<tr>
<td>vendCashDisc</td>
<td>192350044</td>
<td>vendCashDisc</td>
<td></td>
</tr>
<tr>
<td>vendPayment</td>
<td>192350045</td>
<td>vendPayment</td>
<td></td>
</tr>
<tr>
<td>vendInvoiceDisc</td>
<td>192350046</td>
<td>vendInvoiceDisc</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Map Type</td>
<td>Default Value</td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>----------------------------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>vendSettlement</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>crossCompanySettlement</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inventIssueFixedAsset</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>salesRevenue</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>salesConsump</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>salesDisc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>salesCash</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>salesFreight</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>salesFee</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>salesPostage</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>salesRoundOff</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>salesPackingSlip</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>salesOffsetAccountPackingSlip</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>salesIssue</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>salesCommission</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>salesOffsetAccountCommission</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>salesPckSlipRevenue</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>salesPckSlipRevenueOffsetAccount</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rebate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pdsCWLoss</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pdsCWProfit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchConsump</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchDisc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchCash</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchFreight</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchFee</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchPostage</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchOffsetAccount</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchaseInvoiceRoundOff</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchMarkupFreight</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchMarkupCustoms</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchMarkupInsurance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchPckSlp</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchOffsetAccountPckSlp</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>CUSTOMER ENGAGEMENT FIELD</td>
<td>DEFAULT VALUE</td>
<td></td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------------------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>purchReceipt: 192350081</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchStdProfit: 192350082</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchStdLoss: 192350083</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchStdOffsetAccount: 192350084</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inventReceipt: 192350085</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inventIssue: 192350086</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inventProfit: 192350087</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inventLoss: 192350088</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inventStdProfit: 192350089</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inventStdLoss: 192350090</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>opening_ES: 192350091</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchReq: 192350092</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aPInvoice: 192350093</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>budget: 192350094</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchOrderYearEnd: 192350095</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inflationAdjustment_MX: 192350096</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prodReportFinished: 192350097</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prodReportFinishedOffsetAccount: 192350098</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prodIssue: 192350099</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prodIssueOffsetAccount: 192350100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prodReceipt: 192350101</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prodReceiptOffsetAccount: 192350102</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prodPicklistOffsetAccount: 192350103</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prodPicklist: 192350104</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prodWIPValuation: 192350105</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prodWIPIssue: 192350106</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prodWrkCtrIssue: 192350107</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prodScrap: 192350108</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prodScrapOffsetAccount: 192350109</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>projCost: 192350112</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>projPayrollAllocation: 192350113</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>projWIPCostvalue:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-----------------------------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>projOffsetAccountItem</td>
<td>projStatusAccountItem</td>
<td>192350114</td>
<td></td>
</tr>
<tr>
<td>projStatusAccountItem</td>
<td>projTurnover</td>
<td>192350115</td>
<td></td>
</tr>
<tr>
<td>projTurnover</td>
<td>projOnAccount</td>
<td>192350116</td>
<td></td>
</tr>
<tr>
<td>projOnAccount</td>
<td>projSalesvalue</td>
<td>192350117</td>
<td></td>
</tr>
<tr>
<td>projSalesvalue</td>
<td>projSalesvalueOffset</td>
<td>192350118</td>
<td></td>
</tr>
<tr>
<td>projSalesvalueOffset</td>
<td>projAccruedTurnoverProd</td>
<td>192350119</td>
<td></td>
</tr>
<tr>
<td>projAccruedTurnoverProd</td>
<td>projWIPProduction</td>
<td>192350120</td>
<td></td>
</tr>
<tr>
<td>projWIPProduction</td>
<td>projAccruedTurnoverProfit</td>
<td>192350121</td>
<td></td>
</tr>
<tr>
<td>projAccruedTurnoverProfit</td>
<td>projWIPProfit</td>
<td>192350122</td>
<td></td>
</tr>
<tr>
<td>projWIPProfit</td>
<td>projNeverLedger</td>
<td>192350123</td>
<td></td>
</tr>
<tr>
<td>projNeverLedger</td>
<td>projAccruedCost</td>
<td>192350124</td>
<td></td>
</tr>
<tr>
<td>projAccruedCost</td>
<td>projWIPCost</td>
<td>192350125</td>
<td></td>
</tr>
<tr>
<td>projWIPCost</td>
<td>projAccruedRevenueOnAccount</td>
<td>192350126</td>
<td></td>
</tr>
<tr>
<td>projAccruedRevenueOnAccount</td>
<td>projWIPInvoicedOnAccount</td>
<td>192350127</td>
<td></td>
</tr>
<tr>
<td>projWIPInvoicedOnAccount</td>
<td>projNoLedger</td>
<td>192350128</td>
<td></td>
</tr>
<tr>
<td>projNoLedger</td>
<td>payrollDebitAccount</td>
<td>192350129</td>
<td></td>
</tr>
<tr>
<td>payrollDebitAccount</td>
<td>payrollCreditAccount</td>
<td>192350130</td>
<td></td>
</tr>
<tr>
<td>payrollCreditAccount</td>
<td>emplPayment_RU</td>
<td>192350131</td>
<td></td>
</tr>
<tr>
<td>emplPayment_RU</td>
<td>rTSLTranslationDifference</td>
<td>192350132</td>
<td></td>
</tr>
<tr>
<td>rTSLTranslationDifference</td>
<td>rCash</td>
<td>192350133</td>
<td></td>
</tr>
<tr>
<td>rCash</td>
<td>inventRoundingLoss_RU</td>
<td>192350134</td>
<td></td>
</tr>
<tr>
<td>inventRoundingLoss_RU</td>
<td>inventRoundingProfit_RU</td>
<td>192350135</td>
<td></td>
</tr>
<tr>
<td>inventRoundingProfit_RU</td>
<td>advanceAdjustmentGain_RU</td>
<td>192350136</td>
<td></td>
</tr>
<tr>
<td>advanceAdjustmentGain_RU</td>
<td>advanceAdjustmentLoss_RU</td>
<td>192350137</td>
<td></td>
</tr>
<tr>
<td>advanceAdjustmentLoss_RU</td>
<td>fixedAssetsDebit</td>
<td>192350138</td>
<td></td>
</tr>
<tr>
<td>fixedAssetsDebit</td>
<td>fixedAssetsCredit</td>
<td>192350139</td>
<td></td>
</tr>
<tr>
<td>fixedAssetsCredit</td>
<td>cACLedgerJournalNoOff</td>
<td>192350140</td>
<td></td>
</tr>
<tr>
<td>cACLedgerJournalNoOff</td>
<td>amountDiffGain_RU</td>
<td>192350141</td>
<td></td>
</tr>
<tr>
<td>amountDiffGain_RU</td>
<td>amountDiffLoss_RU</td>
<td>192350142</td>
<td></td>
</tr>
<tr>
<td>amountDiffLoss_RU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>192350145</td>
<td>192350146</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>transferGoodsTransit_IN</td>
<td>transferGoodsTransit_IN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350145</td>
<td>: 192350146</td>
<td></td>
<td></td>
</tr>
<tr>
<td>transferScrap_IN</td>
<td>transferScrap_IN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350147</td>
<td>: 192350148</td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchCharge</td>
<td>purchCharge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350149</td>
<td>: 192350149</td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchStockVariation</td>
<td>purchStockVariation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350150</td>
<td>: 192350150</td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchPckSlpPurchaseOffsetAccount</td>
<td>purchPckSlpPurchaseOffsetAccount</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350151</td>
<td>: 192350151</td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchPckSlpTax</td>
<td>purchPckSlpTax</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350152</td>
<td>: 192350152</td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchPckSlpPurchase</td>
<td>purchPckSlpPurchase</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350153</td>
<td>: 192350153</td>
<td></td>
<td></td>
</tr>
<tr>
<td>salesPackingslipTax</td>
<td>salesPackingslipTax</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350154</td>
<td>: 192350154</td>
<td></td>
<td></td>
</tr>
<tr>
<td>projAccruedRevenueSubscription</td>
<td>projAccruedRevenueSubscription</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350155</td>
<td>: 192350155</td>
<td></td>
<td></td>
</tr>
<tr>
<td>projWIPSubscription</td>
<td>projWIPSubscription</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350156</td>
<td>: 192350156</td>
<td></td>
<td></td>
</tr>
<tr>
<td>taxOffsetWithhold_TH</td>
<td>taxOffsetWithhold_TH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350157</td>
<td>: 192350157</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inventStdCostChangeVariance</td>
<td>inventStdCostChangeVariance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350158</td>
<td>: 192350158</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inventSystemRounding</td>
<td>inventSystemRounding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350159</td>
<td>: 192350159</td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchAdvance</td>
<td>purchAdvance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350160</td>
<td>: 192350160</td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchStdCostPurchasePriceVariance</td>
<td>purchStdCostPurchasePriceVariance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350161</td>
<td>: 192350161</td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchAdvanceApplication</td>
<td>purchAdvanceApplication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350162</td>
<td>: 192350162</td>
<td></td>
<td></td>
</tr>
<tr>
<td>prodStdCostProductionVariance</td>
<td>prodStdCostProductionVariance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350163</td>
<td>: 192350163</td>
<td></td>
<td></td>
</tr>
<tr>
<td>salesGoodsInRoute_RU</td>
<td>salesGoodsInRoute_RU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350164</td>
<td>: 192350164</td>
<td></td>
<td></td>
</tr>
<tr>
<td>salesGoodsInRouteOffset_RU</td>
<td>salesGoodsInRouteOffset_RU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350165</td>
<td>: 192350165</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inventInterUnitPayable</td>
<td>inventInterUnitPayable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350166</td>
<td>: 192350166</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inventInterUnitReceivable</td>
<td>inventInterUnitReceivable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350167</td>
<td>: 192350167</td>
<td></td>
<td></td>
</tr>
<tr>
<td>indirectEstimatedAbsorption</td>
<td>indirectEstimatedAbsorption</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350168</td>
<td>: 192350168</td>
<td></td>
<td></td>
</tr>
<tr>
<td>prodStdCostLotSizeVariance</td>
<td>prodStdCostLotSizeVariance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350169</td>
<td>: 192350169</td>
<td></td>
<td></td>
</tr>
<tr>
<td>prodStdCostQuantityVariance</td>
<td>prodStdCostQuantityVariance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350170</td>
<td>: 192350170</td>
<td></td>
<td></td>
</tr>
<tr>
<td>prodStdCostSubstitutionVariance</td>
<td>prodStdCostSubstitutionVariance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350171</td>
<td>: 192350171</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inventStdCostRoundingVariance</td>
<td>inventStdCostRoundingVariance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350172</td>
<td>: 192350172</td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchReceiptFixedAsset</td>
<td>purchReceiptFixedAsset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350173</td>
<td>: 192350173</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pSATransportation</td>
<td>pSATransportation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>: 192350174</td>
<td>: 192350174</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pSACompanyCCClearing</td>
<td>pSACompanyCCClearing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>pSAEmployeeClearing</td>
<td></td>
<td>192350175</td>
<td></td>
</tr>
<tr>
<td>pSAEmployeeAdvance</td>
<td></td>
<td>192350176</td>
<td></td>
</tr>
<tr>
<td>pS writeOffCap</td>
<td></td>
<td>192350177</td>
<td></td>
</tr>
<tr>
<td>pSAProjRetain</td>
<td></td>
<td>192350178</td>
<td></td>
</tr>
<tr>
<td>pSAProjPurchRetain</td>
<td></td>
<td>192350179</td>
<td></td>
</tr>
<tr>
<td>inventStdCostRevaluation</td>
<td></td>
<td>192350180</td>
<td></td>
</tr>
<tr>
<td>purchExpense</td>
<td></td>
<td>192350181</td>
<td></td>
</tr>
<tr>
<td>vat_IN</td>
<td></td>
<td>192350182</td>
<td></td>
</tr>
<tr>
<td>inventMovingAveragePriceDifference</td>
<td></td>
<td>192350183</td>
<td></td>
</tr>
<tr>
<td>salesTax_IN</td>
<td></td>
<td>192350184</td>
<td></td>
</tr>
<tr>
<td>inventMovingAverageCostRevaluation</td>
<td></td>
<td>192350185</td>
<td></td>
</tr>
<tr>
<td>excise_IN</td>
<td></td>
<td>192350186</td>
<td></td>
</tr>
<tr>
<td>intercompanyCost</td>
<td></td>
<td>192350187</td>
<td></td>
</tr>
<tr>
<td>serviceTax_IN</td>
<td></td>
<td>192350188</td>
<td></td>
</tr>
<tr>
<td>intercompanyRevenue</td>
<td></td>
<td>192350189</td>
<td></td>
</tr>
<tr>
<td>customs_IN</td>
<td></td>
<td>192350190</td>
<td></td>
</tr>
<tr>
<td>tDS_IN</td>
<td></td>
<td>192350191</td>
<td></td>
</tr>
<tr>
<td>tCS_IN</td>
<td></td>
<td>192350192</td>
<td></td>
</tr>
<tr>
<td>transferIssue_IN</td>
<td></td>
<td>192350193</td>
<td></td>
</tr>
<tr>
<td>transferReceipt_IN</td>
<td></td>
<td>192350194</td>
<td></td>
</tr>
<tr>
<td>transferProfit_IN</td>
<td></td>
<td>192350195</td>
<td></td>
</tr>
<tr>
<td>transferLoss_IN</td>
<td></td>
<td>192350196</td>
<td></td>
</tr>
<tr>
<td>taxAdjustmentSettlement_IN</td>
<td></td>
<td>192350197</td>
<td></td>
</tr>
<tr>
<td>taxExpense_BR</td>
<td></td>
<td>192350198</td>
<td></td>
</tr>
<tr>
<td>bankStatement</td>
<td></td>
<td>192350199</td>
<td></td>
</tr>
<tr>
<td>emplBalance_RU</td>
<td></td>
<td>192350200</td>
<td></td>
</tr>
<tr>
<td>debitNote_BR</td>
<td></td>
<td>192350201</td>
<td></td>
</tr>
<tr>
<td>custFine_BR</td>
<td></td>
<td>192350202</td>
<td></td>
</tr>
<tr>
<td>vendFine_BR</td>
<td></td>
<td>192350203</td>
<td></td>
</tr>
<tr>
<td>payroll</td>
<td></td>
<td>192350204</td>
<td></td>
</tr>
<tr>
<td>interunitDebit</td>
<td></td>
<td>192350205</td>
<td></td>
</tr>
<tr>
<td>interunitCredit</td>
<td></td>
<td>192350206</td>
<td></td>
</tr>
<tr>
<td>interunitCredit</td>
<td></td>
<td>192350207</td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>fixedAssetsDebit RU</td>
<td></td>
<td>msdyn_defaultcurrency.isocurrencycode</td>
<td></td>
</tr>
<tr>
<td>fixedAssetsCredit RU</td>
<td></td>
<td>msdyn_defaultcurrency.isocurrencycode</td>
<td></td>
</tr>
<tr>
<td>transferInterim_In</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>deferralsDebit RU</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>deferralsCredit RU</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>mCRReturns</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>mCRReturnsConsume</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>mCRUnderpayWriteOff</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>mCRBrokerFee</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>rPayTaxRefundOffset</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>budgetReservation_PSN</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>budgetReservationYearEnd_PSN</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>billSchUnbilledAccountsReceivable</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>billSchAccruedRevenue</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>gST_IN</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>reportingCurrencyAdjustment</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>revRecDeferredRevenue</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>revRecDeferredCostOfGoodsSold</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>revRecPartialRevenue</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>SRUCODE</td>
<td></td>
<td>msdyn_srucode</td>
<td></td>
</tr>
<tr>
<td>revRecDeferredCost</td>
<td></td>
<td>msdyn_srucode</td>
<td></td>
</tr>
<tr>
<td>VALIDATECURRENCY</td>
<td>&gt; &lt;</td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>optional</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>fillin</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>table</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>list</td>
<td></td>
<td>msdyn_validatecurrencycod e</td>
<td></td>
</tr>
<tr>
<td>VALIDATEUSER</td>
<td>&gt; &lt;</td>
<td>msdyn_validateuser</td>
<td></td>
</tr>
<tr>
<td>optional</td>
<td></td>
<td>msdyn_validateuser</td>
<td></td>
</tr>
<tr>
<td>fillin</td>
<td></td>
<td>msdyn_validateuser</td>
<td></td>
</tr>
<tr>
<td>table</td>
<td></td>
<td>msdyn_validateuser</td>
<td></td>
</tr>
<tr>
<td>list</td>
<td></td>
<td>msdyn_validateuser</td>
<td></td>
</tr>
<tr>
<td>DEBITCREDITDEFAULT</td>
<td>&gt; &lt;</td>
<td>msdyn_debitcreditdefault</td>
<td></td>
</tr>
<tr>
<td>none</td>
<td></td>
<td>msdyn_debitcreditdefault</td>
<td></td>
</tr>
<tr>
<td>debit</td>
<td></td>
<td>msdyn_debitcreditdefault</td>
<td></td>
</tr>
<tr>
<td>credit</td>
<td></td>
<td>msdyn_debitcreditdefault</td>
<td></td>
</tr>
<tr>
<td>DEFAULTCURRENCY</td>
<td>=</td>
<td>msdyn_defaultcurrency.isocurrencycode</td>
<td></td>
</tr>
</tbody>
</table>
**Main account categories (msdyn_mainaccountcategories)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAINACCOUNTCATEGORY</td>
<td>=</td>
<td>msdyn_mainaccountcategory</td>
<td></td>
</tr>
<tr>
<td>REFERENCEID</td>
<td>=</td>
<td>msdyn_referenceid</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
<tr>
<td>DISPLAYORDER</td>
<td>=</td>
<td>msdyn_displayorder</td>
<td></td>
</tr>
<tr>
<td>CLOSED</td>
<td>&gt; &lt;</td>
<td>msdyn_closed</td>
<td></td>
</tr>
<tr>
<td>Finance and Operations Field</td>
<td>Map Type</td>
<td>Customer Engagement Column</td>
<td>Default Value</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>-----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td><strong>MAINACCOUNTTYPE</strong></td>
<td>&gt; &lt;</td>
<td>msdyn_mainaccounttypevalue</td>
<td></td>
</tr>
<tr>
<td>blank</td>
<td>192350000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>profitAndLoss</td>
<td>192350001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>revenue</td>
<td>192350002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>expense</td>
<td>192350003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>balanceSheet</td>
<td>192350004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>asset</td>
<td>192350005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>liability</td>
<td>192350006</td>
<td></td>
<td></td>
</tr>
<tr>
<td>equity</td>
<td>192350007</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Mixed reality guides entity (msmrw_guides)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGRATIONKEY</td>
<td>&lt;</td>
<td>msmrw_integrationkey</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>&lt;</td>
<td>msmrw_name</td>
<td></td>
</tr>
<tr>
<td>SCHEMAVERSION</td>
<td>&lt;</td>
<td>msmrw_schemaversion</td>
<td></td>
</tr>
<tr>
<td>GUIDEID</td>
<td>&lt;</td>
<td>msmrw_guideid</td>
<td></td>
</tr>
<tr>
<td>CREATEDON</td>
<td>&lt;</td>
<td>createdon</td>
<td></td>
</tr>
<tr>
<td>LASTMODIFIEDON</td>
<td>&lt;</td>
<td>modifiedon</td>
<td></td>
</tr>
</tbody>
</table>

**Modes of delivery (msdyn_shipvias)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODECODE</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>MODEDESCRIPTION</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
</tbody>
</table>

**Name affixes (msdyn_nameaffixes)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFFIX</td>
<td>=</td>
<td>msdyn_affix</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------</td>
<td>----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>TYPE</td>
<td>&amp;&lt;</td>
<td>msdyn_affixtype</td>
<td></td>
</tr>
<tr>
<td>personalSuffix</td>
<td>806380001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>personalPrefix</td>
<td>806380000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td></td>
<td>msdyn_description</td>
<td></td>
</tr>
</tbody>
</table>

**Operating unit (msdyn_internalorganizations)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LANGUAGEID</td>
<td>&gt;</td>
<td>msdyn_languageid</td>
<td></td>
</tr>
<tr>
<td>NAMEALIAS</td>
<td>&gt;</td>
<td>msdyn_namealias</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>&gt;</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>PARTYNUMBER</td>
<td>&gt;</td>
<td>msdyn_partynumber</td>
<td></td>
</tr>
<tr>
<td>OPERATINGUNITTYPE</td>
<td>&amp;&gt;</td>
<td>msdyn_type</td>
<td></td>
</tr>
<tr>
<td>omDepartment</td>
<td>806380001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>none</td>
<td>806380002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>omCostCenter</td>
<td>806380003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>omValueStream</td>
<td>806380004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>omBusinessUnit</td>
<td>806380005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>omAnyOU</td>
<td>806380006</td>
<td></td>
<td></td>
</tr>
<tr>
<td>retailChannel</td>
<td>192350000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Organization hierarchy - published (msdyn_internalorganizationhierarchies)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>VALIDTO</td>
<td>&gt;</td>
<td>msdyn_validto</td>
<td></td>
</tr>
<tr>
<td>VALIDFROM</td>
<td>&gt;</td>
<td>msdyn_validfrom</td>
<td></td>
</tr>
<tr>
<td>HIERARCHYTYPE</td>
<td>&gt;</td>
<td>msdyn_hierarchytypename</td>
<td></td>
</tr>
<tr>
<td>PARENTORGANIZATIONPARTYNUMBER</td>
<td>&gt;</td>
<td>msdyn_parentpartyid</td>
<td></td>
</tr>
</tbody>
</table>
### Organization hierarchy purposes (msdyn_internalorganizationhierarchypurposes)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIERARCHYTYPE</td>
<td>&gt;</td>
<td>msdyn_hierarchypurposeymename</td>
<td></td>
</tr>
<tr>
<td>HIERARCHYTYPE</td>
<td>&gt;</td>
<td>msdyn_hierarchypurposety_name</td>
<td></td>
</tr>
</tbody>
</table>
### Organization hierarchy type (msdyn_internalorganizationhierarchytypes)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>&gt;</td>
<td>msdyn_name</td>
<td></td>
</tr>
</tbody>
</table>

### Party contacts V3 (msdyn_partyelectronicaddresses)

This template synchronizes data between Finance and Operations apps and Dataverse.
<table>
<thead>
<tr>
<th><strong>FINANCE AND OPERATIONS FIELD</strong></th>
<th><strong>MAP TYPE</strong></th>
<th><strong>CUSTOMER ENGAGEMENT COLUMN</strong></th>
<th><strong>DEFAULT VALUE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNTRYREGIONCODE</td>
<td>=</td>
<td>msdyn_internationalcallingcode</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
<tr>
<td>ELECTRONICADDRESSID</td>
<td>=</td>
<td>msdyn_electronicaddressnumber</td>
<td></td>
</tr>
<tr>
<td>PARTYNUMBER</td>
<td>=</td>
<td>msdyn_partyid.msdyn_partynumber</td>
<td></td>
</tr>
<tr>
<td>LOCATOR</td>
<td>=</td>
<td>msdyn_locator</td>
<td></td>
</tr>
<tr>
<td>LOCATOREXTENSION</td>
<td>=</td>
<td>msdyn_locatorextension</td>
<td></td>
</tr>
<tr>
<td>ISINSTANTMESSAGE</td>
<td>&gt; &lt;</td>
<td>msdyn_isinstantmessage</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISMOBILEPHONE</td>
<td>&gt; &lt;</td>
<td>msdyn_ismobile</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISPRIMARY</td>
<td>&gt; &lt;</td>
<td>msdyn_isprimary</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISPRIVATE</td>
<td>&gt; &lt;</td>
<td>msdyn_isprivate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PURPOSE</td>
<td>=</td>
<td>msdyn_purpose</td>
<td></td>
</tr>
<tr>
<td>TYPE</td>
<td>&gt; &lt;</td>
<td>msdyn_type</td>
<td></td>
</tr>
<tr>
<td></td>
<td>phone : 192350001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>email : 192350002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>url : 192350003</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>telex : 192350004</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>fax : 192350005</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>facebook : 192350006</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>twitter : 192350007</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>linkedin : 192350008</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Payment day lines CDS V2 (msdyn_paymentdaylines)**

This template synchronizes data between Finance and Operations apps and Dataverse.
<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linenumber</td>
<td>=</td>
<td>msdyn_linenumber</td>
<td></td>
</tr>
<tr>
<td>Frequency</td>
<td>&lt;</td>
<td>msdyn_frequency</td>
<td></td>
</tr>
<tr>
<td>Dayofweek</td>
<td>&lt;</td>
<td>msdyn_dayofweek</td>
<td></td>
</tr>
<tr>
<td>Dayofmonth</td>
<td>=</td>
<td>msdyn_dayofmonth</td>
<td></td>
</tr>
</tbody>
</table>

**Payment days CDS (msdyn_paymentdays)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
</tbody>
</table>

**Payment schedule (msdyn_paymentschedules)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
<tr>
<td>AllocationMethod</td>
<td>&lt;</td>
<td>msdyn_allocationmethod</td>
<td></td>
</tr>
<tr>
<td>PaymentFrequencyUnits</td>
<td>&lt;</td>
<td>msdyn_paymentfrequencyunit</td>
<td></td>
</tr>
<tr>
<td>PaymentFrequency</td>
<td>=</td>
<td>msdyn_paymentfrequency</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------</td>
<td>-----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>NUMBEROFPAYMENTS</td>
<td>=</td>
<td>msdyn_numberofpayments</td>
<td></td>
</tr>
<tr>
<td>FIXEDPAYMENTAMOUNT</td>
<td>=</td>
<td>msdyn_fixedpaymentamount</td>
<td></td>
</tr>
<tr>
<td>MINIMUMPAYMENTAMOUNT</td>
<td>=</td>
<td>msdyn_minimumpaymentamount</td>
<td></td>
</tr>
<tr>
<td>SALESTAXALLOCATIONMETHOD</td>
<td>&gt; &lt;</td>
<td>msdyn_salestaxallocationmethod</td>
<td></td>
</tr>
<tr>
<td></td>
<td>proportional: 806380000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>firstRate: 806380001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lastRate: 806380002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOTES</td>
<td>=</td>
<td>msdyn_note</td>
<td></td>
</tr>
</tbody>
</table>

**Payment schedule lines (msdyn_paymentschedulelines)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAYMENTSCHEDULENAME</td>
<td>=</td>
<td>msdyn_paymentschedulename</td>
<td></td>
</tr>
<tr>
<td>LINENUMBER</td>
<td>=</td>
<td>msdyn_linenumber</td>
<td></td>
</tr>
<tr>
<td>PERIODS AFTER DUE DATE</td>
<td>=</td>
<td>msdyn_periodsofterdue</td>
<td></td>
</tr>
<tr>
<td>PERCENT OR AMOUNT</td>
<td>&gt; &lt;</td>
<td>msdyn_percentoramount</td>
<td></td>
</tr>
<tr>
<td></td>
<td>percent: 806380000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>amount: 806380001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PERCENT OR AMOUNT VALUE</td>
<td>=</td>
<td>msdyn_percentoramountvalue</td>
<td></td>
</tr>
</tbody>
</table>

**Personal character types (msdyn_personalcharactertypes)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE DESCRIPTION</td>
<td>=</td>
<td>msdyn_typedescription</td>
<td></td>
</tr>
<tr>
<td>TYPENAME</td>
<td>=</td>
<td>msdyn_typename</td>
<td></td>
</tr>
</tbody>
</table>

**Position type (cdm_positiontypes)**

This template synchronizes data between Finance and Operations apps and Dataverse.
<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSITIONTYPEID</td>
<td>=</td>
<td>cdm_name</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>cdm_description</td>
<td></td>
</tr>
<tr>
<td>CLASSIFICATION</td>
<td>&gt; &lt;</td>
<td></td>
<td>cdm_classification</td>
</tr>
<tr>
<td></td>
<td>fulltime</td>
<td>754400000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>parttime</td>
<td>754400001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>blank</td>
<td>754400002</td>
<td></td>
</tr>
</tbody>
</table>

**Position worker assignments (cdm_positionworkerassignmentmaps)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERSONNELNUMBER</td>
<td>=</td>
<td>cdm_workerid.cdm_workerid</td>
<td></td>
</tr>
<tr>
<td>POSITIONID</td>
<td>=</td>
<td>cdm_jobpositionid.cdm_job</td>
<td></td>
</tr>
<tr>
<td>VALIDFROM</td>
<td>=</td>
<td>cdm_validfrom</td>
<td></td>
</tr>
<tr>
<td>VALIDTO</td>
<td>=</td>
<td>cdm_validto</td>
<td></td>
</tr>
</tbody>
</table>

**Positions V2 (cdm_jobpositions)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSITIONID</td>
<td>=</td>
<td>cdm_jobpositionnumber</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>cdm_description</td>
<td></td>
</tr>
<tr>
<td>ACTIVATION</td>
<td>=</td>
<td>cdm_activation</td>
<td></td>
</tr>
<tr>
<td>AVAILABLEFORASSIGNMENT</td>
<td>=</td>
<td>cdm_availableforassignment</td>
<td></td>
</tr>
<tr>
<td>FULLTIMEEQUIVALENT</td>
<td>=</td>
<td>cdm_fulltimeequivalent</td>
<td></td>
</tr>
<tr>
<td>DETAILEFFECTIVE</td>
<td>=</td>
<td>cdm_validfrom</td>
<td></td>
</tr>
<tr>
<td>DETAILEXPIRATION</td>
<td>=</td>
<td>cdm_validto</td>
<td></td>
</tr>
<tr>
<td>RETIREMENT</td>
<td>=</td>
<td>cdm_retirement</td>
<td></td>
</tr>
<tr>
<td>POSITIONTYPEID</td>
<td>=</td>
<td>cdm_positiontypeid.cdm_name</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------</td>
<td>-----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>JOBID</td>
<td>=</td>
<td>cdm_jobid.cdm_name</td>
<td></td>
</tr>
</tbody>
</table>

**Price customer groups (msdyn_pricecustomergroups)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUPCODE</td>
<td>=</td>
<td>msdyn_groupcode</td>
<td></td>
</tr>
<tr>
<td>GROUPNAME</td>
<td>=</td>
<td>msdyn_groupname</td>
<td></td>
</tr>
</tbody>
</table>

**Product Number Identified Barcode (msdyn_productbarcodes)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCTNUMBER</td>
<td>&gt;</td>
<td>msdyn_productnumberid.msdyn_productnumber</td>
<td></td>
</tr>
<tr>
<td>BARCODE</td>
<td>&gt;</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>BARCODE</td>
<td>&gt;</td>
<td>msdyn_barcode</td>
<td></td>
</tr>
<tr>
<td>PRODUCTQUANTITY</td>
<td>&gt;</td>
<td>msdyn_productquantity</td>
<td></td>
</tr>
<tr>
<td>PRODUCTDESCRIPTION</td>
<td>&gt;</td>
<td>msdyn_productdescription</td>
<td></td>
</tr>
<tr>
<td>BARCODESETUPID</td>
<td>&gt;</td>
<td>msdyn_barcodesetupid</td>
<td></td>
</tr>
<tr>
<td>PRODUCTQUANTITYUNITS</td>
<td>&gt;</td>
<td>msdyn_unitofmeasureid.msdyn_symbol</td>
<td></td>
</tr>
<tr>
<td>ISDEFAULTSCANNEDBARCODE</td>
<td>&gt;&gt;</td>
<td>msdyn_isdefaultscannedbarcode</td>
<td></td>
</tr>
<tr>
<td>ISDEFAULTPRINTEDBARCODE</td>
<td>&gt;&gt;</td>
<td>msdyn_isdefaultprintedbarcode</td>
<td></td>
</tr>
<tr>
<td>ISDEFAULTDISPLAYEDBARCODE</td>
<td>&gt;&gt;</td>
<td>msdyn_isdefaultdisplayedbarcode</td>
<td></td>
</tr>
</tbody>
</table>

**Product categories (msdyn_productcategories)**

This template synchronizes data between Finance and Operations apps and Dataverse.
<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCTCATEGORYHIERARCHYNAME</td>
<td>=</td>
<td>msdyn_hierarchy.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>ISCATEGORYINHERITINGPARENTPRODUCTATTRIBUTES</td>
<td>&gt; &lt;</td>
<td>msdyn_isinheritingparentproductattributes</td>
<td></td>
</tr>
<tr>
<td>PROJECTCATEGORYNAME</td>
<td>=</td>
<td>msdyn_projectcategoryname</td>
<td></td>
</tr>
<tr>
<td>ISTANGIBLEPRODUCT</td>
<td>&gt; &lt;</td>
<td>msdyn_istangibleproduct</td>
<td></td>
</tr>
<tr>
<td>ISCATEGORYINHERITINGPARENTCATEGORYATTRIBUTES</td>
<td>&gt; &lt;</td>
<td>msdyn_isinheritingparentcategoryattributes</td>
<td></td>
</tr>
<tr>
<td>CATEGORYCODE</td>
<td>=</td>
<td>msdyn_code</td>
<td></td>
</tr>
<tr>
<td>CATEGORYDESCRIPTION</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
<tr>
<td>CATEGORYKEYWORDS</td>
<td>=</td>
<td>msdyn_keywords</td>
<td></td>
</tr>
<tr>
<td>CATEGORYNAME</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>FRIENDLYCATEGORYNAME</td>
<td>=</td>
<td>msdyn_friendlycategoryname</td>
<td></td>
</tr>
<tr>
<td>PARENTPRODUCTCATEGORYNAME</td>
<td>=</td>
<td>msdyn_parentproductcategory.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>PARENTPRODUCTCATEGORYHIERARCHYNAME</td>
<td>&gt;</td>
<td>msdyn_parentproductcategory.msdyn_hierarchy.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>CATEGORYRECORDID</td>
<td>&gt;</td>
<td>msdyn_externalproductcategoryid</td>
<td></td>
</tr>
<tr>
<td>EXTERNALID</td>
<td>=</td>
<td>msdyn_integrationid</td>
<td></td>
</tr>
</tbody>
</table>

**Product category assignments (msdyn_productcategoryassignments)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCTNUMBER</td>
<td>=</td>
<td>msdyn_globalproduct.msdyn_productnumber</td>
<td></td>
</tr>
<tr>
<td>PRODUCTCATEGORYNAME</td>
<td>=</td>
<td>msdyn_productcategory.msdyn_name</td>
<td></td>
</tr>
</tbody>
</table>
### Product category hierarchies (msdyn_productcategoryhierarchies)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HierarchyName</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>HierarchyDescription</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
</tbody>
</table>

### Product category hierarchy roles (msdyn_productcategoryhierarchyroles)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProductCategoryHierarchyName</td>
<td>=</td>
<td>msdyn_hierarchy.msdyn_name</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HierarchyRole</th>
</tr>
</thead>
<tbody>
<tr>
<td>procurement : 192350000</td>
</tr>
<tr>
<td>sales : 192350001</td>
</tr>
<tr>
<td>retail : 192350002</td>
</tr>
<tr>
<td>commodity : 192350003</td>
</tr>
<tr>
<td>financials : 192350004</td>
</tr>
<tr>
<td>retailSpecialGroup : 192350005</td>
</tr>
<tr>
<td>retailVendorProductHierarchy : 192350006</td>
</tr>
<tr>
<td>retailChannelNavigation : 192350007</td>
</tr>
<tr>
<td>packingMaterials_W : 192350008</td>
</tr>
<tr>
<td>commonDataService : 192350009</td>
</tr>
<tr>
<td>costManagement : 192350010</td>
</tr>
<tr>
<td>engineeringProduct : 192350011</td>
</tr>
</tbody>
</table>

### Product default order settings V2 (msdyn_productspecificdefaultordersettings)

This template synchronizes data between Finance and Operations apps and Dataverse.
<table>
<thead>
<tr>
<th>FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVENTORYWAREHOUSEID</td>
<td>=</td>
<td>msdyn_inventorywarehouse.msdyn_warehouseidentifier</td>
<td></td>
</tr>
<tr>
<td>INVENTORIESITEID</td>
<td>=</td>
<td>msdyn_inventorysite.msdyn_sited</td>
<td></td>
</tr>
<tr>
<td>INVENTORYATPDELAYEDDE MANDOFFSETDAYS</td>
<td>=</td>
<td>msdyn_inventoryatpdelayed_demandoffsetdays</td>
<td></td>
</tr>
<tr>
<td>INVENTORYATPDELAYEDSUPPLYOFFSETDAYS</td>
<td>=</td>
<td>msdyn_inventoryatpdelayed_supplyoffsetdays</td>
<td></td>
</tr>
<tr>
<td>ITEMNUMBER</td>
<td>=</td>
<td>msdyn_itemnumber.msdyn_itemnumber</td>
<td></td>
</tr>
<tr>
<td>INVENTORYATPBACKWARD DEMANDTIMEFENCEDAYS</td>
<td>=</td>
<td>msdyn_inventoryatpbackward_demandtimefencedays</td>
<td></td>
</tr>
<tr>
<td>INVENTORYATPBACKWARD SUPPLYTIMEFENCEDAYS</td>
<td>=</td>
<td>msdyn_inventoryatpbackwardsupplytimefencedays</td>
<td></td>
</tr>
<tr>
<td>INVENTARYATPTIMEFENCEDAYS</td>
<td>=</td>
<td>msdyn_inventoryatptimefencedays</td>
<td></td>
</tr>
<tr>
<td>MAXIMUMINVENTORYORDERQUANTITY</td>
<td>=</td>
<td>msdyn_maximuminventoryorderquantity</td>
<td></td>
</tr>
<tr>
<td>MAXIMUMPROCUREMENTORDERQUANTITY</td>
<td>=</td>
<td>msdyn_maximumprocurementorderquantity</td>
<td></td>
</tr>
<tr>
<td>MAXIMUMSALESORDERQUANTITY</td>
<td>=</td>
<td>msdyn_maximumsalesorderquantity</td>
<td></td>
</tr>
<tr>
<td>MINIMUMINVENTORYORDERQUANTITY</td>
<td>=</td>
<td>msdyn_minimuminventoryorderquantity</td>
<td></td>
</tr>
<tr>
<td>MINIMUMPROCUREMENTORDERQUANTITY</td>
<td>=</td>
<td>msdyn_minimumprocurementorderquantity</td>
<td></td>
</tr>
<tr>
<td>MINIMUMSALESORDERQUANTITY</td>
<td>=</td>
<td>msdyn_minimumsalesorderquantity</td>
<td></td>
</tr>
<tr>
<td>STANDARDINVENTORYORDERQUANTITY</td>
<td>=</td>
<td>msdyn_standardinventoryorderquantity</td>
<td></td>
</tr>
<tr>
<td>STANDARDPROCUREMENTORDERQUANTITY</td>
<td>=</td>
<td>msdyn_standardprocurementorderquantity</td>
<td></td>
</tr>
<tr>
<td>STANDARDSALESORDERQUANTITY</td>
<td>=</td>
<td>msdyn_standardsalesorderquantity</td>
<td></td>
</tr>
<tr>
<td>INVENTORYLEADTIMEDAYS</td>
<td>=</td>
<td>msdyn_inventoryleadtimedays</td>
<td></td>
</tr>
<tr>
<td><strong>FIELD</strong></td>
<td><strong>MAP TYPE</strong></td>
<td><strong>CUSTOMER ENGAGEMENT COLUMN</strong></td>
<td><strong>DEFAULT VALUE</strong></td>
</tr>
<tr>
<td>---------------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>INVENTORYQUANTITYMULTIPLES</td>
<td>=</td>
<td>msdyn_inventoryquantitymultiples</td>
<td></td>
</tr>
<tr>
<td>PROCUREMENTQUANTITYMULTIPLES</td>
<td>=</td>
<td>msdyn PROCUREMENTquantitymultiples</td>
<td></td>
</tr>
<tr>
<td>SALESQUANTITYMULTIPLES</td>
<td>=</td>
<td>msdyn_salesquantitymultiples</td>
<td></td>
</tr>
<tr>
<td>PROCUREMENTSITEID</td>
<td>=</td>
<td>msdyn.procurementsite.msdyn_siteid</td>
<td></td>
</tr>
<tr>
<td>PROCUREMENTLEADTIMEDAYS</td>
<td>=</td>
<td>msdyn.procurementleadtimedays</td>
<td></td>
</tr>
<tr>
<td>SALESSITEID</td>
<td>=</td>
<td>msdyn.salesite.msdyn_siteid</td>
<td></td>
</tr>
<tr>
<td>SALESATPDELAYEDDEMANDOFFSETDAYS</td>
<td>=</td>
<td>msdyn_salesatpdelayeddemandoffsetdays</td>
<td></td>
</tr>
<tr>
<td>SALESATPDELAYEDSUPPLYOFFSETDAYS</td>
<td>=</td>
<td>msdyn_salesatpdelayedsupplyoffsetdays</td>
<td></td>
</tr>
<tr>
<td>SALESATPBACKWARDDEMANDTIMEFENCEDAYS</td>
<td>=</td>
<td>msdyn_salesatpbackwarddemandtimefencedays</td>
<td></td>
</tr>
<tr>
<td>SALESATPBACKWARDSUPPLYTIMEFENCEDAYS</td>
<td>=</td>
<td>msdyn_salesatpbackwardsupplytimefencedays</td>
<td></td>
</tr>
<tr>
<td>SALESATPTIMEFENCEDAYS</td>
<td>=</td>
<td>msdyn_salesatptimefencedays</td>
<td></td>
</tr>
<tr>
<td>SALESLEADTIMEDAYS</td>
<td>=</td>
<td>msdyn_salesleadtimedays</td>
<td></td>
</tr>
<tr>
<td>PROCUREMENTWAREHOUSEID</td>
<td>=</td>
<td>msdyn.procurementwarehouse.msdyn_warehouseidentifier</td>
<td></td>
</tr>
<tr>
<td>SALESWAREHOUSEID</td>
<td>=</td>
<td>msdyn_saleswarehouse.msdyn_warehouseidentifier</td>
<td></td>
</tr>
<tr>
<td>AREINVENTORYDEFAULTORDERSETTINGSOVERRIDDEN</td>
<td>&gt; &lt;</td>
<td>msdyn_areinventoryorderdefaultsettingsoverridden</td>
<td></td>
</tr>
</tbody>
</table>

- **MAP TYPE** represents the mapping type used in the system.
- **CUSTOMER ENGAGEMENT COLUMN** indicates the column name where the data is stored.
- **DEFAULT VALUE** specifies the default value for the field.
<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISINVENTORYATPINCLUDINGPLANNEDORDERS</td>
<td></td>
<td>msdyn_isinventoryatpincludingplannedorders</td>
<td>false: False true: True</td>
</tr>
<tr>
<td>ISINVENTORYUSINGWORKINGDAYS</td>
<td></td>
<td>msdyn_isinventoryusingworkingdays</td>
<td>no: False yes: True</td>
</tr>
<tr>
<td>ISINVENTORYSITEMANDATORY</td>
<td></td>
<td>msdyn_isinventorysitemandatory</td>
<td>no: False yes: True</td>
</tr>
<tr>
<td>ISINVENTORYPROCESSINGSTOPPED</td>
<td></td>
<td>msdyn_isinventoryprocessingstopped</td>
<td>no: False yes: True</td>
</tr>
<tr>
<td>ISPROCUREMENTUSINGWORKINGDAYS</td>
<td></td>
<td>msdyn_isprocurementusingworkingdays</td>
<td>no: False yes: True</td>
</tr>
<tr>
<td>ISPROCUREMENTSITEMANDATORY</td>
<td></td>
<td>msdyn_isprocurementsitemandatory</td>
<td>no: False yes: True</td>
</tr>
<tr>
<td>ISPROCUREMENTPROCESSINGSTOPPED</td>
<td></td>
<td>msdyn_isprocurementprocessingstopped</td>
<td>no: False yes: True</td>
</tr>
<tr>
<td>ARESALESDEFAULTORDERSSETTINGSOVERRIDDEN</td>
<td></td>
<td>msdyn_areasalesorderdefaultsoverridden</td>
<td>no: False yes: True</td>
</tr>
<tr>
<td>ISSALESATPINCLUDINGPLANNEDORDERS</td>
<td></td>
<td>msdyn_isalesatpincludingplannedorders</td>
<td>false: False true: True</td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>----------</td>
<td>-----------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>ISSALESITEMMANDATORY</td>
<td>$&lt;,$</td>
<td>msdyn_issalessitemandatory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISSALESLEADTIMEOVERRIDED</td>
<td>$&lt;,$</td>
<td>msdyn_issalessleadtimeoverridden</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISSALESPROCESSINGSTOPPED</td>
<td>$&lt;,$</td>
<td>msdyn_issalessprocessingstopped</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISINVENTORYWAREHOUSEMANDATORY</td>
<td>$&lt;,$</td>
<td>msdyn_isinventorywarehousemandatory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISPROCUREMENTWAREHOUSEMANDATORY</td>
<td>$&lt;,$</td>
<td>msdyn_isprocurementwarehousemandatory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISSALESWAREHOUSEMANDATORY</td>
<td>$&lt;,$</td>
<td>msdyn_issalesswholesalemandatory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPERATIONALSITEID</td>
<td>=</td>
<td>msdyn_operationalsite.msdyn_siteid</td>
<td></td>
</tr>
<tr>
<td>PRODUCTCOLORID</td>
<td>=</td>
<td>msdyn_productcolor.msdyn_productcolorname</td>
<td></td>
</tr>
<tr>
<td>PRODUCTCONFIGURATIONID</td>
<td>=</td>
<td>msdyn_productconfiguration.msdyn_productconfiguration</td>
<td></td>
</tr>
<tr>
<td>PRODUCTSIZEID</td>
<td>=</td>
<td>msdyn_productsize.msdyn_productsize</td>
<td></td>
</tr>
<tr>
<td>PRODUCTSTYLEID</td>
<td>=</td>
<td>msdyn_productstyle.msdyn_productstyle</td>
<td></td>
</tr>
</tbody>
</table>

**Product dimension groups (msdyn_productdimensiongroups)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>WILLSALESPRICESEARCHUSEPRODUCTSTYLE</td>
<td>$&lt;,$</td>
<td>msdyn_willsalespricesearchuseproductstyle</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : true</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FIELD</strong></td>
<td><strong>MAP TYPE</strong></td>
<td><strong>CUSTOMER ENGAGEMENT COLUMN</strong></td>
<td><strong>DEFAULT VALUE</strong></td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>---------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>WILLPURCHASEPRICESEARCHUSEPRODUCTSIZE</td>
<td>&gt; &lt;</td>
<td>msdyn_willpurchasepricesearchuseproductsize</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WILLSALESPRICESSEARCHUSEPRODUCTCONFIGURATION</td>
<td>&gt; &lt;</td>
<td>msdyn_willsalespricesearchuseprodconfig</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WILLSALESPRICESSEARCHUSEPRODUCTCOLOR</td>
<td>&gt; &lt;</td>
<td>msdyn_willsalespricesearchuseproductcolor</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WILLPURCHASEPRICESEARCHUSEPRODUCTSTYLE</td>
<td>&gt; &lt;</td>
<td>msdyn_willpurchasepricesearchuseproductstyle</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WILLPURCHASEPRICESEARCHUSEPRODUCTCONFIGURATION</td>
<td>&gt; &lt;</td>
<td>msdyn_willpurchasepricesearchuseprodconfig</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WILLSALESPRICESSEARCHUSEPRODUCTCOLOR</td>
<td>&gt; &lt;</td>
<td>msdyn_willsalespricesearchuseproductcolor</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISPRODUCTSTYLEACTIVE</td>
<td>&gt; &lt;</td>
<td>msdyn_isproductstyleactive</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISPRODUCTSIZEACTIVE</td>
<td>&gt; &lt;</td>
<td>msdyn_isproductsizeactive</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISPRODUCTCONFIGURATIONACTIVE</td>
<td>&gt; &lt;</td>
<td>msdyn_isproductconfigurationactive</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISPRODUCTCOLORACTIVE</td>
<td>&gt; &lt;</td>
<td>msdyn_isproductcoloractive</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GROUPNAME</td>
<td>=</td>
<td>msdyn_groupname</td>
<td></td>
</tr>
<tr>
<td>GROUPDESCRIPTION</td>
<td>=</td>
<td>msdyn_groupdescription</td>
<td></td>
</tr>
<tr>
<td>PRODUCTVARIANTNOMENCLATURENAME</td>
<td>=</td>
<td>msdyn_productvariantnomenclaturename</td>
<td></td>
</tr>
<tr>
<td>WILLSALESPRICESSEARCHUSEPRODUCTSIZE</td>
<td>&gt; &lt;</td>
<td>msdyn_willsalespricesearchuseproductsize</td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : true</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : false</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Product master colors (msdyn_sharedproductcolors)
This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCTCOLORID</td>
<td>&gt;</td>
<td>msdyn_productcolor.msdyn_productcolorname</td>
<td></td>
</tr>
<tr>
<td>PRODUCTMASTERNUMBER</td>
<td>&gt;</td>
<td>msdyn_globalproduct.msdyn_productnumber</td>
<td></td>
</tr>
<tr>
<td>REPLENISHMENTWEIGHT</td>
<td>&gt;</td>
<td>msdyn_replenishmentweight</td>
<td></td>
</tr>
<tr>
<td>DISPLAYSEQUENCENUMBER</td>
<td>&gt;</td>
<td>msdyn_displaysequencenumber</td>
<td></td>
</tr>
</tbody>
</table>

**Product master configurations (msdyn_sharedproductconfigurations)**
This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTAINERUNITSYMBOL</td>
<td>&gt;</td>
<td>msdyn_containerunit.msdyn_symbol</td>
<td></td>
</tr>
<tr>
<td>PRODUCTCONFIGURATIONID</td>
<td>&gt;</td>
<td>msdyn_productconfiguration.msdyn_productconfiguration</td>
<td></td>
</tr>
<tr>
<td>PRODUCTMASTERNUMBER</td>
<td>&gt;</td>
<td>msdyn_globalproduct.msdyn_productnumber</td>
<td></td>
</tr>
<tr>
<td>REPLENISHMENTWEIGHT</td>
<td>&gt;</td>
<td>msdyn_replenishmentweight</td>
<td></td>
</tr>
<tr>
<td>DISPLAYSEQUENCENUMBER</td>
<td>&gt;</td>
<td>msdyn_displaysequencenumber</td>
<td></td>
</tr>
</tbody>
</table>

**Product master sizes (msdyn_sharedproductsizes)**
This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCTMASTERNUMBER</td>
<td>&gt;</td>
<td>msdyn_globalproduct.msdyn_productnumber</td>
<td></td>
</tr>
<tr>
<td>PRODUCTSIZEID</td>
<td>&gt;</td>
<td>msdyn_productsize.msdyn_productsize</td>
<td></td>
</tr>
<tr>
<td>REPLENISHMENTWEIGHT</td>
<td>&gt;</td>
<td>msdyn_replenishmentweight</td>
<td></td>
</tr>
<tr>
<td>DISPLAYSEQUENCENUMBER</td>
<td>&gt;</td>
<td>msdyn_displaysequencenumber</td>
<td></td>
</tr>
</tbody>
</table>

This template synchronizes data between Finance and Operations apps and Dataverse.
**Product master styles (msdyn_sharedproductstyles)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCTMASTERNUMBER</td>
<td>&gt;</td>
<td>msdyn_globalproduct.msdy n_productnumber</td>
<td></td>
</tr>
<tr>
<td>PRODUCTSTYLEID</td>
<td>&gt;</td>
<td>msdyn_productstyle.msdyn _productstyle</td>
<td></td>
</tr>
<tr>
<td>REPLENISHMENTWEIGHT</td>
<td>&gt;</td>
<td>msdyn_replenishmentweight</td>
<td></td>
</tr>
<tr>
<td>DISPLAYSEQUENCENUMBER</td>
<td>&gt;</td>
<td>msdyn_displaysequencenu mber</td>
<td></td>
</tr>
</tbody>
</table>

**Product receipt header (msdyn_purchaseorderreceipts)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECORDID</td>
<td>&gt;</td>
<td>msdyn_recordid</td>
<td></td>
</tr>
<tr>
<td>PRODUCTRECEIPTDATE</td>
<td>&gt;</td>
<td>msdyn_datereceived</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSLATITUDE</td>
<td>&gt;</td>
<td>msdyn_deliveryaddresslatitu de</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSLONGITUDE</td>
<td>&gt;</td>
<td>msdyn_deliveryaddresslongi tude</td>
<td></td>
</tr>
<tr>
<td>PURCHASEORDERNUMBER</td>
<td>&gt;</td>
<td>msdyn_purchaseorder.msdy n_name</td>
<td></td>
</tr>
<tr>
<td>DELIVERYMODEID</td>
<td>&gt;</td>
<td>msdyn_shipvia.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>DELIVERYTERMSID</td>
<td>&gt;</td>
<td>msdyn_deliveryterm.msdyn _termscode</td>
<td></td>
</tr>
<tr>
<td>ORDervendorACCOUNTNUMBER</td>
<td>&gt;</td>
<td>msdyn_ordervendor.msdyn_ vendoraccountnumber</td>
<td></td>
</tr>
<tr>
<td>REQUESTERPERSONNELNUMBER</td>
<td>&gt;</td>
<td>msdyn_requesterpersonnel. cdm_workernumber</td>
<td></td>
</tr>
<tr>
<td>ISDELIVERYADDRESSPRIVATE</td>
<td>&gt;&gt;</td>
<td>msdyn_isdeliveryaddresspri vate</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSCOUNTRYREGIONID</td>
<td>&gt;</td>
<td>msdyn_deliveryaddresscoun tryregionid</td>
<td></td>
</tr>
<tr>
<td><strong>FIELD</strong></td>
<td><strong>MAP TYPE</strong></td>
<td><strong>CUSTOMER ENGAGEMENT COLUMN</strong></td>
<td><strong>DEFAULT VALUE</strong></td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------------</td>
<td>------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>DELIVERYADDRESSCOUNTYID</td>
<td>&gt;</td>
<td>msdyn_deliveryaddresscountyid</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSSTATEID</td>
<td>&gt;</td>
<td>msdyn_deliveryaddressstateid</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSZIPCODE</td>
<td>&gt;</td>
<td>msdyn_deliveryaddresszipcode</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSNAME</td>
<td>&gt;</td>
<td>msdyn_deliveryaddressname</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSTIMEZONE</td>
<td>&gt;</td>
<td>msdyn_deliveryaddresstimezone</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSPOSTBOX</td>
<td>&gt;</td>
<td>msdyn_deliveryaddresspostbox</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSSTREETNUMBER</td>
<td>&gt;</td>
<td>msdyn_deliveryaddressstreetnumber</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSSTREET</td>
<td>&gt;</td>
<td>msdyn_deliveryaddressstreet</td>
<td></td>
</tr>
<tr>
<td>PRODUCTRECEIPTNUMBER</td>
<td>&gt;</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>ATTENTIONINFORMATION</td>
<td>&gt;</td>
<td>msdyn_note</td>
<td></td>
</tr>
<tr>
<td>DELIVERYCITYINKANA</td>
<td>&gt;</td>
<td>msdyn_deliverycityinkana</td>
<td></td>
</tr>
<tr>
<td>DELIVERYSTREETINKANA</td>
<td>&gt;</td>
<td>msdyn_deliverystreetinkana</td>
<td></td>
</tr>
<tr>
<td>FORMATTEDDELIVERYADDRESS</td>
<td>&gt;</td>
<td>msdyn_formatteddeliveryaddress</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSLOCATIONID</td>
<td>&gt;</td>
<td>msdyn_deliveryaddresslocationid</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSCITY</td>
<td>&gt;</td>
<td>msdyn_deliveryaddresscity</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSDESCRIPTION</td>
<td>&gt;</td>
<td>msdyn_deliveryaddressdescription</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSDISTRICTNAME</td>
<td>&gt;</td>
<td>msdyn_deliveryaddressdistricname</td>
<td></td>
</tr>
<tr>
<td>DELIVERYBUILDINGCOMPLIMENT</td>
<td>&gt;</td>
<td>msdyn_deliverybuildingcompliment</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSDUNSNUMBER</td>
<td>&gt;</td>
<td>msdyn_deliveryaddressdunsnumber</td>
<td></td>
</tr>
</tbody>
</table>

*Product receipt line (msdyn_purchaseorderreceiptproducts)*
This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECORDID</td>
<td>&gt;</td>
<td>msdyn_recordid</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSCOUNTR YREGIONID</td>
<td>&gt;</td>
<td>msdyn_deliveryaddresscountryregionid</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSCOUNTY ID</td>
<td>&gt;</td>
<td>msdyn_deliveryaddresscountryid</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSSTATEID</td>
<td>&gt;</td>
<td>msdyn_deliveryaddressstateid</td>
<td></td>
</tr>
<tr>
<td>EXPECTEDDELIERYDATE</td>
<td>&gt;</td>
<td>msdyn_expecteddeliverydate</td>
<td></td>
</tr>
<tr>
<td>EXTERNALITEMNUMBER</td>
<td>&gt;</td>
<td>msdyn_externalitemnumber</td>
<td></td>
</tr>
<tr>
<td>ITEMBATCHNUMBER</td>
<td>&gt;</td>
<td>msdyn_itembatchnumber</td>
<td></td>
</tr>
<tr>
<td>ITEMSERIALNUMBER</td>
<td>&gt;</td>
<td>msdyn_itemserialnumber</td>
<td></td>
</tr>
<tr>
<td>LINEDESCRIPTION</td>
<td>&gt;</td>
<td>msdyn_linedescription</td>
<td></td>
</tr>
<tr>
<td>ORDEREDPURCHASEQUANTITY</td>
<td>&gt;</td>
<td>msdyn_orderedpurchasequantity</td>
<td></td>
</tr>
<tr>
<td>PROCUREMENTPRODUCTCATEGORYNAME</td>
<td>&gt;</td>
<td>msdyn_procurementproductcategory.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>PROCUREMENTPRODUCTCATEGORYHIERARCHYNAME</td>
<td>&gt;</td>
<td>msdyn_procurementproductcategory.msdyn_hierarchy.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>PRODUCTRECEIPTDATE</td>
<td>&gt;</td>
<td>msdyn_productreceiptdate</td>
<td></td>
</tr>
<tr>
<td>PRODUCTRECEIPTHEAORDERRECORDID</td>
<td>&gt;</td>
<td>msdyn_purchaseorderreceipt.msdyn_recordid</td>
<td></td>
</tr>
<tr>
<td>PRODUCTRECEIPTNUMBER</td>
<td>&gt;</td>
<td>msdyn_productreceiptnumber</td>
<td></td>
</tr>
<tr>
<td>PURCHASEORDERLINENUMBER</td>
<td>&gt;</td>
<td>msdyn_purchaseorderproduct.msdyn_lineorder</td>
<td></td>
</tr>
<tr>
<td>PURCHASEORDERNUMBER</td>
<td>&gt;</td>
<td>msdyn_purchaseorderproduct.msdyn_purchaseorder.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>PURCHASEORDERNUMBER</td>
<td>&gt;</td>
<td>msdyn_purchaseorder.msdyn_name</td>
<td></td>
</tr>
</tbody>
</table>
### Finance and Operations Field | Map Type | Customer Engagement Column | Default Value
--- | --- | --- | ---
PURCHASEUNITSYMBOL | > | msdyn_purchaseunitsymbol.msdyn_symbol | 
RECEIVEDINVENTORYQUANTITY | > | msdyn_receivedinventoryquantity | 
RECEIVEDINVENTORYSTATUSID | > | msdyn_receivedinventorystatusid | 
RECEIVEDPURCHASEQUANTITY | > | msdyn_quantity | 
RECEIVINGSITEID | > | msdyn_receivingsiteid.msdyn_siteid | 
RECEIVINGWAREHOUSEID | > | msdyn_associatetowarehouse.msdyn_warehouseidentifier | 
RECEIVINGWAREHOUSELOCATIONID | > | msdyn_receivingwarehouselocation.msdyn_warehouselocationid | 
RECEIVINGWAREHOUSEID | > | msdyn_receivingwarehouselocation.msdyn_warehouse. msdyn_warehouseidentifier | 
REMAININGINVENTORYQUANTITY | > | msdyn_remaininginventoryquantity | 
REMAININGPURCHASEQUANTITY | > | msdyn_remainingpurchasequantity | 
PRODUCTNUMBER | > | msdyn_product.msdyn_productnumber | 

**Product specific unit conversions (msdyn_productspecificunitofmeasureconversions)**

This template synchronizes data between Finance and Operations apps and Dataverse.

### Finance and Operations Field | Map Type | Customer Engagement Column | Default Value
--- | --- | --- | ---
DENOMINATOR | = | msdyn_denominator | 
NUMERATOR | = | msdyn_numerator | 
FACTOR | = | msdyn_factor | 
FROMUNITSYMBOL | = | msdyn_fromunit.msdyn_symbol | 
TOUNITSYMBOL | = | msdyn_tounit.msdyn_symbol |
### Roundings

<table>
<thead>
<tr>
<th>FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCTNUMBER</td>
<td>=</td>
<td>msdyn_globalproduct.msdyn_productnumber</td>
<td></td>
</tr>
<tr>
<td>INNEROFFSET</td>
<td>=</td>
<td>msdyn_inneroffset</td>
<td></td>
</tr>
<tr>
<td>OUTEROFFSET</td>
<td>=</td>
<td>msdyn_outeroffset</td>
<td></td>
</tr>
<tr>
<td>ROUNING</td>
<td>&gt; &lt;</td>
<td></td>
<td>msdyn_rounding</td>
</tr>
<tr>
<td></td>
<td>nearest</td>
<td>192350000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>up</td>
<td>192350001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>down</td>
<td>192350002</td>
<td></td>
</tr>
</tbody>
</table>

### Prospects (leads)

This template synchronizes data between Finance and Operations apps and Dataverse.

Source filter: (IsB2BProspect = 1)

Reversed source filter: (msdyn_b2bcommerceprospect eq true)

<table>
<thead>
<tr>
<th>FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROSPECTNAME</td>
<td>=</td>
<td>fullname</td>
<td></td>
</tr>
<tr>
<td>PROSPECTPARTYNUMBER</td>
<td>=</td>
<td>msdyn_partyid.msdyn_partynumber</td>
<td></td>
</tr>
<tr>
<td>PROSPECTID</td>
<td>=</td>
<td>msdyn_prospectid</td>
<td></td>
</tr>
<tr>
<td>PRIMARYPHONENUMBER</td>
<td>=</td>
<td>telephone1</td>
<td></td>
</tr>
<tr>
<td>PRIMARYPHONENUMBER_EXTENSION</td>
<td>=</td>
<td>msdyn_telephone1extension</td>
<td></td>
</tr>
<tr>
<td>PRIMARYPHONENUMBER_DESCRIPTION</td>
<td>=</td>
<td>description</td>
<td></td>
</tr>
<tr>
<td>PRIMARYEMAILADDRESS</td>
<td>=</td>
<td>emailaddress1</td>
<td></td>
</tr>
<tr>
<td>PRIMARYEMAILADDRESS_DESCRIPTION</td>
<td>=</td>
<td>msdyn_emailaddress1description</td>
<td></td>
</tr>
<tr>
<td>PRIMARYURL</td>
<td>=</td>
<td>websiteurl</td>
<td></td>
</tr>
<tr>
<td>PRIMARYURLDESCRIPTION</td>
<td>=</td>
<td>msdyn_websiteurldescription</td>
<td></td>
</tr>
<tr>
<td>ADDRESSCITY</td>
<td>=</td>
<td>address1_city</td>
<td></td>
</tr>
<tr>
<td>ADDRESSCOUNTRYREGION</td>
<td>=</td>
<td>address1_country</td>
<td></td>
</tr>
<tr>
<td>ADDRESISCOCODE</td>
<td>=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------</td>
<td>-----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>ADDRESSCOUNTYID</td>
<td>=</td>
<td>address1_county</td>
<td></td>
</tr>
<tr>
<td>ADDRESSTYPEID</td>
<td>=</td>
<td>address1_stateorprovince</td>
<td></td>
</tr>
<tr>
<td>ADDRESSSTREET</td>
<td>=</td>
<td>address1_line1</td>
<td></td>
</tr>
<tr>
<td>ADDRESSZIPCODE</td>
<td>=</td>
<td>address1_postalcode</td>
<td></td>
</tr>
<tr>
<td>ADDRESSPOSTBOX</td>
<td>=</td>
<td>address1_postofficebox</td>
<td></td>
</tr>
<tr>
<td>ADDRESSLOCATIONROLES</td>
<td>&lt;=</td>
<td>none</td>
<td>Business</td>
</tr>
<tr>
<td>none</td>
<td>=&gt;</td>
<td>address1_addresstypecode</td>
<td>1</td>
</tr>
<tr>
<td>COMPANYNAME</td>
<td>=</td>
<td>companyname</td>
<td></td>
</tr>
<tr>
<td>COMPANYSIZE</td>
<td>=</td>
<td>numberofemployees</td>
<td></td>
</tr>
<tr>
<td>ADDRESSTLATITUDE</td>
<td>=</td>
<td>address1_latitude</td>
<td></td>
</tr>
<tr>
<td>ADDRESSLONGITUDE</td>
<td>=</td>
<td>address1_longitude</td>
<td></td>
</tr>
<tr>
<td>CURRENCYCODE</td>
<td>=</td>
<td>transactioncurrencyid.isocurrencycode</td>
<td></td>
</tr>
<tr>
<td>CUSTOMERGROUPID</td>
<td>=</td>
<td>msdyn_customergroupid.msdyn_groupid</td>
<td></td>
</tr>
<tr>
<td>SALESTAXGROUPCODE</td>
<td>=</td>
<td>msdyn_salestaxgroup.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>PRIMARYFAXNUMBER</td>
<td>=</td>
<td>fax</td>
<td></td>
</tr>
<tr>
<td>PRIMARYFAXNUMBERDESCRIPTION</td>
<td>=</td>
<td>msdyn_faxdescription</td>
<td></td>
</tr>
<tr>
<td>PRIMARYFAXNUMBEREXTENSION</td>
<td>=</td>
<td>msdyn_faxextension</td>
<td></td>
</tr>
<tr>
<td>RETAILCHANNELOPERATINGUNITPARTYNUMBER</td>
<td>=</td>
<td>msdyn_retailchannel.msdyn_partynumber</td>
<td></td>
</tr>
<tr>
<td>ISB2BPROSPECT</td>
<td>&lt;=</td>
<td>msdyn_b2bcommerceprospect</td>
<td></td>
</tr>
<tr>
<td>no : False</td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B2BPROSPECTSTATUS</td>
<td>&lt;=</td>
<td>msdyn_b2bprospectstatus</td>
<td></td>
</tr>
<tr>
<td>Pending : 192350000</td>
<td>Approved : 192350001</td>
<td>Rejected : 192350002</td>
<td></td>
</tr>
</tbody>
</table>

Purchase order header document attachments (annotations)
This template synchronizes data between Finance and Operations apps and Dataverse.

Source filter: ((DocumentAttachmentTypeCode == "Note") || (DocumentAttachmentTypeCode == "URL"))

Reversed source filter: objecttypecode eq 'msdyn_purchaseorder'

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOCUMENTID</td>
<td>=</td>
<td>msdyn_notesid</td>
<td></td>
</tr>
<tr>
<td>NOTES</td>
<td>=</td>
<td>notetext</td>
<td></td>
</tr>
<tr>
<td>ATTACHMENTDESCRIPTION</td>
<td>=</td>
<td>subject</td>
<td></td>
</tr>
<tr>
<td>PURCHASEORDERNUMBER</td>
<td>=</td>
<td>msdyn_relatedentityid</td>
<td></td>
</tr>
<tr>
<td>DOCUMENTATTACHMENTTYPECODE</td>
<td>&lt; &lt;</td>
<td>none</td>
<td>Note</td>
</tr>
<tr>
<td>none</td>
<td>&gt; &gt;</td>
<td>objecttypecode</td>
<td>msdyn_purchaseorder</td>
</tr>
<tr>
<td>ACCESSRESTRICTION</td>
<td>&gt; &lt;</td>
<td>msdyn_restriction</td>
<td></td>
</tr>
<tr>
<td></td>
<td>internal: 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>external: 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Purchase order headers V2 (msdyn_purchaseorders)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELIVERYADDRESSNAME</td>
<td>&gt;</td>
<td>msdyn_addressname</td>
<td></td>
</tr>
<tr>
<td>REQUESTEDDELIVERYDATE</td>
<td>=</td>
<td>msdyn_dateexpected</td>
<td></td>
</tr>
<tr>
<td>PURCHASEORDERNUMBER</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>PAYMENTTERMSNAME</td>
<td>=</td>
<td>msdyn_paymentterm.msdy n_name</td>
<td></td>
</tr>
<tr>
<td>DEFAULTRECEIVINGWAREHOUSEID</td>
<td>=</td>
<td>msdyn_receivetowarehouse.msdyn_warehouseidentifier</td>
<td></td>
</tr>
<tr>
<td>ACCOUNTINGDATE</td>
<td>=</td>
<td>msdyn_accountingdate</td>
<td></td>
</tr>
<tr>
<td>AREPRICESINCLUDINGSALESTAX</td>
<td>&gt; &lt;</td>
<td>msdyn_arepriceincludingsalesestax</td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATTENTIONINFORMATION</td>
<td>&gt;</td>
<td>msdyn_attentioninformation</td>
<td></td>
</tr>
<tr>
<td>CASHDISCOUNTCODE</td>
<td>=</td>
<td>msdyn_cashdiscountcode</td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------</td>
<td>-----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>CASHDISCOUNTPERCENTAGE</td>
<td>=</td>
<td>msdyn_cashdiscountpercentage</td>
<td></td>
</tr>
<tr>
<td>CONTACTPERSONID</td>
<td>=</td>
<td>msdyn_contactpersonid.msdyn_contactpersonid</td>
<td></td>
</tr>
<tr>
<td>CURRENCYCODE</td>
<td>=</td>
<td>transactioncurrencyid.isocurrencycode</td>
<td></td>
</tr>
<tr>
<td>DEFAULTRECEIVINGSITEID</td>
<td>=</td>
<td>msdyn_defaultreceivingsiteid.msdyn_siteid</td>
<td></td>
</tr>
<tr>
<td>DELIVERYTERMSID</td>
<td>=</td>
<td>msdyn_deliveryterm.msdyn_termscode</td>
<td></td>
</tr>
<tr>
<td>EMAIL</td>
<td>=</td>
<td>msdyn_email</td>
<td></td>
</tr>
<tr>
<td>ISCHANGEMANAGEMENTACTIVE</td>
<td>&gt;&gt;</td>
<td>msdyn_ischangemanagementactive</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISDELIVEREDDIRECTLY</td>
<td>&gt;&gt;</td>
<td>msdyn_isdeliverydirectly</td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------</td>
<td>-----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>LANGUAGEID</td>
<td></td>
<td>msdyn_language</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&gt; &lt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ar       : 192350000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ar-ae    : 192350001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>cs       : 192350002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>da       : 192350003</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>de       : 192350004</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>de-at    : 192350005</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>de-ch    : 192350006</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>en-au    : 192350007</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>en-gb    : 192350009</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>en-ie    : 192350010</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>en-in    : 192350011</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>en-ca    : 192350008</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>en-my    : 192350012</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>en-nz    : 192350013</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>en-sg    : 192350014</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>en-us    : 192350015</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>en-za    : 192350016</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>es       : 192350017</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>es-mx    : 192350018</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>et       : 192350019</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>fi       : 192350020</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>fr       : 192350021</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>fr-be    : 192350022</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>fr-ca    : 192350023</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>fr-ch    : 192350024</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>hu       : 192350025</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>is       : 192350026</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>it       : 192350027</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>it-ch    : 192350028</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ja       : 192350029</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lt       : 192350030</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lv       : 192350031</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>nb-no    : 192350032</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>nl       : 192350033</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>nl-be    : 192350034</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pl       : 192350035</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pt-br    : 192350036</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ru       : 192350037</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sv       : 192350038</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>th       : 192350039</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tr       : 192350040</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>zh-hans  : 192350041</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORDERERPERSONNELNUMBER</td>
<td>=</td>
<td>msdyn_ordererpersonnelnumb er.cdm_workernumber</td>
<td></td>
</tr>
<tr>
<td>REASONCODE</td>
<td>=</td>
<td>msdyn_reasoncode</td>
<td></td>
</tr>
<tr>
<td>REASONCOMMENT</td>
<td>=</td>
<td>msdyn_reasoncomment</td>
<td></td>
</tr>
<tr>
<td>TOTALDISCOUNTPERCENTAGE</td>
<td>=</td>
<td>msdyn_totaldiscountpercentage</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT FIELD</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>URL</td>
<td>=</td>
<td>msdyn_url</td>
<td></td>
</tr>
<tr>
<td>VENDORORDERREFERENCE</td>
<td>=</td>
<td>msdyn_vendororderreferenc e</td>
<td></td>
</tr>
<tr>
<td>VENDORPAYMENTMETHODDNAME</td>
<td>=</td>
<td>msdyn_vendorpaymentmet hod .msdyn_name</td>
<td></td>
</tr>
<tr>
<td>VENDORPAYMENTMETHODDSPECIFICATIONNAME</td>
<td>&gt;</td>
<td>msdyn_vendorpaymentmet hodspecificationname</td>
<td></td>
</tr>
<tr>
<td>ORDERVENDORACCOUNTNUMBER</td>
<td>=</td>
<td>msdyn_vendor.accountnum ber</td>
<td></td>
</tr>
<tr>
<td>DELIVERYMODEID</td>
<td>=</td>
<td>msdyn_shipvia.msdyn_name e</td>
<td></td>
</tr>
<tr>
<td>INVOICEVENDORACCOUNTNUMBER</td>
<td>=</td>
<td>msdyn_invoicevendoraccount.accountnumber</td>
<td></td>
</tr>
<tr>
<td>DOCUMENTAPPROVALSTATUS</td>
<td>&gt;&gt;</td>
<td>msdyn_documentapprovals tatus</td>
<td></td>
</tr>
<tr>
<td></td>
<td>draft : 192350000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>inReview : 192350001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>rejected : 192350002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>approved : 192350003</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>inExternalReview : 192350004</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>finalized : 192350005</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>confirmed : 192350006</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PURCHASEORDERSTATUS</td>
<td>&gt;&gt;</td>
<td>msdyn_purchaseorderstatus</td>
<td></td>
</tr>
<tr>
<td></td>
<td>none : 192350000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>backorder : 192350001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>received : 192350002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>invoiced : 192350003</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>canceled : 192350004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSCITY</td>
<td>&gt;</td>
<td>msdyn_city</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSCOUNTRYREGIONISOCODE</td>
<td>&gt;</td>
<td>msdyn_country</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSSTATEID</td>
<td>&gt;</td>
<td>msdyn_stateorprovince</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSZIPCODE</td>
<td>&gt;</td>
<td>msdyn_postalcode</td>
<td></td>
</tr>
<tr>
<td>DELIVERYADDRESSSTREET</td>
<td>&gt;</td>
<td>msdyn_address1</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSSTREETNUMBER</td>
<td>&gt;</td>
<td>msdyn_invoiceaddressstreet number</td>
<td></td>
</tr>
<tr>
<td>Finance and Operations Field</td>
<td>Map Type</td>
<td>Customer Engagement Column</td>
<td>Default Value</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>-----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Confirmed Delivery Date</td>
<td>=</td>
<td>msdyn_confirmeddeliverydate</td>
<td></td>
</tr>
<tr>
<td>Delivery Address Longitude</td>
<td>&gt;</td>
<td>msdyn_longitude</td>
<td></td>
</tr>
<tr>
<td>Delivery Address Latitude</td>
<td>&gt;</td>
<td>msdyn_latitude</td>
<td></td>
</tr>
<tr>
<td>Delivery Address Country ID</td>
<td>&gt;</td>
<td>msdyn_deliveryaddresscountryid</td>
<td></td>
</tr>
<tr>
<td>Delivery Address County ID</td>
<td>&gt;</td>
<td>msdyn_deliveryaddresscountryid</td>
<td></td>
</tr>
<tr>
<td>Delivery Address Description</td>
<td>&gt;</td>
<td>msdyn_deliveryaddressdescription</td>
<td></td>
</tr>
<tr>
<td>Delivery Address District Name</td>
<td>&gt;</td>
<td>msdyn_deliveryaddressdistrictname</td>
<td></td>
</tr>
<tr>
<td>Delivery Address DUNS Number</td>
<td>&gt;</td>
<td>msdyn_deliveryaddressdunsnumber</td>
<td></td>
</tr>
<tr>
<td>Delivery Address Location ID</td>
<td>&gt;</td>
<td>msdyn_deliveryaddresslocationid</td>
<td></td>
</tr>
<tr>
<td>Delivery Address Postbox</td>
<td>&gt;</td>
<td>msdyn_deliveryaddresspostbox</td>
<td></td>
</tr>
<tr>
<td>Delivery Address Time Zone</td>
<td>&gt;</td>
<td>msdyn_deliveryaddresstimezone</td>
<td></td>
</tr>
<tr>
<td>Delivery Building Compliment</td>
<td>&gt;</td>
<td>msdyn_deliverybuildingcompliment</td>
<td></td>
</tr>
<tr>
<td>Formatted Delivery Address</td>
<td>&gt;</td>
<td>msdyn_formatteddeliveryaddress</td>
<td></td>
</tr>
<tr>
<td>Formatted Invoice Address</td>
<td>&gt;</td>
<td>msdyn_formattedinvoiceaddress</td>
<td></td>
</tr>
<tr>
<td>Invoice Address City</td>
<td>&gt;</td>
<td>msdyn_invoiceaddresscity</td>
<td></td>
</tr>
<tr>
<td>Invoice Address Country ID</td>
<td>&gt;</td>
<td>msdyn_invoiceaddresscountryid</td>
<td></td>
</tr>
<tr>
<td>Invoice Address Country</td>
<td>&gt;</td>
<td>msdyn_invoiceaddresscountry</td>
<td></td>
</tr>
<tr>
<td>Invoice Address State</td>
<td>&gt;</td>
<td>msdyn_invoiceaddressstate</td>
<td></td>
</tr>
</tbody>
</table>
### Released products V2 (msdyn_sharedproductdetails)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th><strong>FINANCE AND OPERATIONS FIELD</strong></th>
<th><strong>MAP TYPE</strong></th>
<th><strong>CUSTOMER ENGAGEMENT COLUMN</strong></th>
<th><strong>DEFAULT VALUE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCTNUMBER</td>
<td>&gt;</td>
<td>msdyn_globalproduct.msdyn_productnumber</td>
<td></td>
</tr>
<tr>
<td>INTRASTATCHARGEPERCENTAGE</td>
<td>&gt;</td>
<td>msdyn_intrastatchargepercentage</td>
<td></td>
</tr>
<tr>
<td>ITEMNUMBER</td>
<td>&gt;</td>
<td>msdyn_itemnumber</td>
<td></td>
</tr>
<tr>
<td>APPROXIMATESALESTAXPERCENTAGE</td>
<td>&gt;</td>
<td>msdyn_approximatesalestaxpercentage</td>
<td></td>
</tr>
<tr>
<td>BESTBEFOREPERIODDAYS</td>
<td>&gt;</td>
<td>msdyn_bestbeforeperioddays</td>
<td></td>
</tr>
<tr>
<td>CARRYINGCOSTABCCODE</td>
<td>&gt;&gt;</td>
<td>msdyn_carryingcostabcode</td>
<td></td>
</tr>
<tr>
<td>CONSTANTSCRAPQUANTITY</td>
<td>&gt;</td>
<td>msdyn_constantscrapquantity</td>
<td></td>
</tr>
<tr>
<td>COSTCHARGESQUANTITY</td>
<td>&gt;</td>
<td>msdyn_costchargesquantity</td>
<td></td>
</tr>
<tr>
<td>DEFAULTRECEIVINGQUANTITY</td>
<td>&gt;</td>
<td>msdyn_defaultreceivingquantity</td>
<td></td>
</tr>
<tr>
<td>FIXEDPURCHASEPRICECHARGES</td>
<td>&gt;</td>
<td>msdyn_fixedpurchasepricecharges</td>
<td></td>
</tr>
<tr>
<td>FIXEDSALESPRICECHARGES</td>
<td>&gt;</td>
<td>msdyn_fixedsalespricecharges</td>
<td></td>
</tr>
<tr>
<td>GROSSDEPTH</td>
<td>&gt;</td>
<td>msdyn_grossdepth</td>
<td></td>
</tr>
<tr>
<td>GROSSPRODUCTHEIGHT</td>
<td>&gt;</td>
<td>msdyn_grossproductheight</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Map Type</td>
<td>Customer Engagement Column</td>
<td>Default Value</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>GrossProductWidth</td>
<td>&gt;</td>
<td>msdyn_grossproductwidth</td>
<td></td>
</tr>
<tr>
<td>InventoryUnitsSymbol</td>
<td>&gt;</td>
<td>msdyn_inventoryunitsymbol</td>
<td>msdyn_symbol</td>
</tr>
<tr>
<td>IsDiscountPosRegistrationProhibited</td>
<td>&gt;&gt;</td>
<td>msdyn_isdiscountposregistr</td>
<td>ationprohibited</td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsExemptFromAutomaticNotificationAndCance</td>
<td>&gt;&gt;</td>
<td>msdyn_exemptautomaticnotifi</td>
<td>cationcancel</td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsInstallmentEligible</td>
<td>&gt;&gt;</td>
<td>msdyn_isinstallmenteligible</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsInterCompanyPurchaseUsageBlocked</td>
<td>&gt;&gt;</td>
<td>msdyn_isintercompanypurcha</td>
<td>haseusageblocked</td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsInterCompanySalesUsageBlocked</td>
<td>&gt;&gt;</td>
<td>msdyn_isintercompanysales</td>
<td>usageblocked</td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsManualDiscountPosRegistrationProhibited</td>
<td>&gt;&gt;</td>
<td>msdyn_ismanualdiscposregis</td>
<td>trationprohibited</td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsPhantom</td>
<td>&gt;&gt;</td>
<td>msdyn_isphantom</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsPosRegistrationBlocked</td>
<td>&gt;&gt;</td>
<td>msdyn_isposregistrationblo</td>
<td>cked</td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsPosRegistrationQuantityNegative</td>
<td>&gt;&gt;</td>
<td>msdyn_isposregistrationqua</td>
<td>ntitynegative</td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsPurchasePriceAutomaticallyUpdated</td>
<td>&gt;&gt;</td>
<td>msdyn_ispurchasepriceauto</td>
<td>maticallyupdated</td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsPurchasePriceIncludingCharges</td>
<td>&gt;&gt;</td>
<td>msdyn_ispurchasepriceinclu</td>
<td>dingcharges</td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsSalesWithholdingTaxCalculated</td>
<td>&gt;&gt;</td>
<td>msdyn_issaleswithholdingtax</td>
<td>calculated</td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Map Type</td>
<td>Customer Engagement Field</td>
<td>Default Value</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------------</td>
<td>---------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>IsRestrictedForCoupons</td>
<td>&gt;&gt;</td>
<td>msdyn_isrestrictedforcoupons</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsSalesPriceAdjustmentAllowed</td>
<td>&gt;&gt;</td>
<td>msdyn_issalespriceadjustmentallowed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsSalesPriceIncludingCharges</td>
<td>&gt;&gt;</td>
<td>msdyn_issalespriceincludingcharges</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsScaleProduct</td>
<td>&gt;&gt;</td>
<td>msdyn_isscaleproduct</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsShipAloneEnabled</td>
<td>&gt;&gt;</td>
<td>msdyn_isshipaloneenabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsUnitCostProductVariantSpecific</td>
<td>&gt;&gt;</td>
<td>msdyn_isunitcostproductvariantspecific</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsVariantShelfLabelPrintingEnabled</td>
<td>&gt;&gt;</td>
<td>msdyn_isvariantshelflabelsprintingenabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IsZeroPricePosRegistrationAllowed</td>
<td>&gt;&gt;</td>
<td>msdyn_iszeropriceposregistrationallowed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KeyInPriceRequirementsATPosRegister</td>
<td>&gt;&gt;</td>
<td>msdyn_keyinpricerequirementsatposregister</td>
<td></td>
</tr>
<tr>
<td></td>
<td>notMandatory : 806380000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>newPrice : 806380001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>higherEqual : 806380002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lowerEqual : 806380003</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>noPrice : 806380004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KeyInQuantityRequirementsATPosRegister</td>
<td>&gt;&gt;</td>
<td>msdyn_keyinquantityrequirementsatposregister</td>
<td></td>
</tr>
<tr>
<td></td>
<td>notMandatory : 806380000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>keyIn : 806380001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>notKeyIn : 806380002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MarginABCCode</td>
<td>&gt;&gt;</td>
<td>msdyn_marginabccode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>none : 806380000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>a : 806380001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>b : 806380002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>c : 806380003</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FINANCE AND OPERATIONS FIELD</strong></td>
<td><strong>MAP TYPE</strong></td>
<td><strong>CUSTOMER ENGAGEMENT FIELD</strong></td>
<td><strong>DEFAULT VALUE</strong></td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------</td>
<td>--------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>MAXIMUMPICKQUANTITY</td>
<td>&gt;</td>
<td>msdyn_maximumpickquantity</td>
<td></td>
</tr>
<tr>
<td>MUSTKEYINCOMMENTATPOSREGISTER</td>
<td>&gt;&gt;</td>
<td>msdyn_mustkeyincommentatposregister</td>
<td>False, True</td>
</tr>
</tbody>
</table>
| NECESSARYPRODUCTION             | >           | msdyn_necessaryproductio       | workingtimeschedul
| WORKINGTIMESCHEDULINGPROPERTYID |             |                                 |                   |
| NETPRODUCTWEIGHT                | >           | msdyn_netproductweight         |                   |
| PACKINGDUTYQUANTITY             | >           | msdyn_packingdutyquantity      |                   |
| POSREGISTRATIONACTIVATIONDATE   | >           | msdyn_posregistrationactivatio |                   |
| POSREGISTRATIONBLOKEDDATE       | >           | msdyn_posregistrationblockedd |                   |
| POSREGISTRATIONPLANENDEDDATE    | >           | msdyn_posregistrationplanen    |                   |
| POTENCYBASEATTRIBUTEVALUE       | >           | msdyn_potencybaseattributeval |                   |
| POTENCYBASEATTRIBUTEVALUEENTRYEVENT | >>          | msdyn_potencybaseattributeval | purchProdReceipt:
<p>|                                  |             |                                 | 806380000          |
|                                  |             |                                 | quality: 806380001 |
| PRODUCTTYPE                     | &gt;&gt;          | msdyn_producttype              |                   |
|                                 |             |                                 | item: 806380001    |
|                                 |             |                                 | service: 806380002 |
| PRODUCTIONCONSUMPTIONDENSITYCONVERSIONFACTOR | &gt;          | msdyn_productionconsump          |                   |
|                                 |             | tiondensityconversion         |                   |
| PRODUCTIONCONSUMPTIONDEPTHCONVERSIONFACTOR | &gt;          | msdyn_productionconsump          |                   |
|                                 |             | tiondepthconversion           |                   |
| PRODUCTIONCONSUMPTIONHEIGHTCONVERSIONFACTOR | &gt;          | msdyn_productionconsump          |                   |
|                                 |             | tionheightconversion         |                   |
| PRODUCTIONCONSUMPTIONWIDTHCONVERSIONFACTOR | &gt;          | msdyn_productionconsump          |                   |
|                                 |             | tionwidthconversion           |                   |</p>
<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCTVOLUME</td>
<td>&gt;</td>
<td>msdyn_productvolume</td>
<td></td>
</tr>
<tr>
<td>PURCHASECHARGESQUANTITY</td>
<td>&gt;</td>
<td>msdyn_purchasechargesquantity</td>
<td></td>
</tr>
<tr>
<td>PURCHASEOVERDELIVERYPERCENTAGE</td>
<td>&gt;</td>
<td>msdyn_purchaseoverdeliverypercentage</td>
<td></td>
</tr>
<tr>
<td>PURCHASEPRICE</td>
<td>&gt;</td>
<td>msdyn_purchaseprice</td>
<td></td>
</tr>
<tr>
<td>PURCHASEPRICEDATE</td>
<td>&gt;</td>
<td>msdyn_purchasepricedate</td>
<td></td>
</tr>
<tr>
<td>PURCHASEPRICINGPRECISION</td>
<td>&gt;</td>
<td>msdyn_purchasepricingprecision</td>
<td></td>
</tr>
<tr>
<td>PURCHASEUNDERDELIVERYPERCENTAGE</td>
<td>&gt;</td>
<td>msdyn_purchaseunderdeliverypercentage</td>
<td></td>
</tr>
<tr>
<td>RAWMATERIALPICKINGPRINCIPLE</td>
<td>&gt;&gt;</td>
<td>msdyn_rawmaterialpickingprinciple</td>
<td></td>
</tr>
<tr>
<td>SALESCHARGESQUANTITY</td>
<td>&gt;</td>
<td>msdyn_saleschargesquantity</td>
<td></td>
</tr>
<tr>
<td>SALESOVERDELIVERYPERCENTAGE</td>
<td>&gt;</td>
<td>msdyn_salesoverdeliverypercentage</td>
<td></td>
</tr>
<tr>
<td>SALESPRICE</td>
<td>&gt;</td>
<td>msdyn_salesprice</td>
<td></td>
</tr>
<tr>
<td>SALESPRICECALCULATIONCHARGESPERCENTAGE</td>
<td>&gt;</td>
<td>msdyn_salespricecalculationchargespercentage</td>
<td></td>
</tr>
<tr>
<td>SALESPRICECALCULATIONCONTRIBUTIONRATIO</td>
<td>&gt;</td>
<td>msdyn_salespricecalculationcontributionratio</td>
<td></td>
</tr>
<tr>
<td>SALESPRICECALCULATIONMODEL</td>
<td>&gt;&gt;</td>
<td>msdyn_salespricecalculationmodel</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[none: 806380000; contributionratio: 806380001; percentMarkup: 806380002]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SALESPRICEDATE</td>
<td>&gt;</td>
<td>msdyn_salespricedate</td>
<td></td>
</tr>
<tr>
<td>SALESPRICINGPRECISION</td>
<td>&gt;</td>
<td>msdyn_salespricingprecision</td>
<td></td>
</tr>
<tr>
<td>SALESOVERDELIVERYPERCENTAGE</td>
<td>&gt;</td>
<td>msdyn_salesunderdeliverypercentage</td>
<td></td>
</tr>
<tr>
<td>Finance and Operations Field</td>
<td>Map Type</td>
<td>Customer Engagement Field</td>
<td>Default Value</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>---------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>SalesUnitSymbol</td>
<td>&gt;</td>
<td>msdyn_salesunitsymbol</td>
<td></td>
</tr>
<tr>
<td>ScaleIndicator</td>
<td>&gt;&gt;</td>
<td>msdyn_scaleindicator</td>
<td></td>
</tr>
<tr>
<td>SellStartDate</td>
<td>&gt;</td>
<td>msdyn_sellstartdate</td>
<td></td>
</tr>
<tr>
<td>ShelfAdvicePeriodDays</td>
<td>&gt;</td>
<td>msdyn_shelfadviceperioddays</td>
<td></td>
</tr>
<tr>
<td>ShelfLifePeriodDays</td>
<td>&gt;</td>
<td>msdyn_shelflifeperioddays</td>
<td></td>
</tr>
<tr>
<td>ShipStartDate</td>
<td>&gt;</td>
<td>msdyn_shipstartdate</td>
<td></td>
</tr>
<tr>
<td>TareProductWeight</td>
<td>&gt;</td>
<td>msdyn_tareproductweight</td>
<td></td>
</tr>
<tr>
<td>TransferOrderOverDeliveryPercentage</td>
<td>&gt;</td>
<td>msdyn_transferorderoverdeliverypercentage</td>
<td></td>
</tr>
<tr>
<td>TransferOrderUnderDeliveryPercentage</td>
<td>&gt;</td>
<td>msdyn_transferorderunderdeliverypercentage</td>
<td></td>
</tr>
<tr>
<td>UnitCost</td>
<td>&gt;</td>
<td>msdyn_unitcost</td>
<td></td>
</tr>
<tr>
<td>UnitCostDate</td>
<td>&gt;</td>
<td>msdyn_unitcostdate</td>
<td></td>
</tr>
<tr>
<td>UnitCostQuantity</td>
<td>&gt;</td>
<td>msdyn_unitcostquantity</td>
<td></td>
</tr>
<tr>
<td>VariablesScraperPercentage</td>
<td>&gt;</td>
<td>msdyn_variablesscrapercentage</td>
<td></td>
</tr>
<tr>
<td>WarehouseMobileDeviceEDescriptionLine1</td>
<td>&gt;</td>
<td>msdyn_warehousemobiledevicedescriptionline1</td>
<td></td>
</tr>
<tr>
<td>WarehouseMobileDeviceEDescriptionLine2</td>
<td>&gt;</td>
<td>msdyn_warehousemobiledevicedescriptionline2</td>
<td></td>
</tr>
<tr>
<td>WillInventoryIssueAutomaticallyReportAsFinished</td>
<td>&gt;&gt;</td>
<td>msdyn_willinventoryissueautomaticallyreportasfinished</td>
<td></td>
</tr>
<tr>
<td>WillInventoryReceiptIgnoreFlushingPrinciple</td>
<td>&gt;&gt;</td>
<td>msdyn_willinventoryreceiptignoreflushing</td>
<td></td>
</tr>
<tr>
<td>WillPickingWorkbenchApplyBoxingLogic</td>
<td>&gt;&gt;</td>
<td>msdyn_willpickingworkbenchapplyboxinglogic</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------</td>
<td>-----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>WILLTOTALPURCHASEDISCOUNTCALCULATIONINCLUDPRODUCT</td>
<td>&gt;&gt; no: False yes: True</td>
<td>msdyn_willtotalpurchdiscountcalcincludeproduct</td>
<td></td>
</tr>
<tr>
<td>WILLTOTALSALESDISCOUNTCALCULATIONINCLUDPRODUCT</td>
<td>&gt;&gt; no: False yes: True</td>
<td>msdyn_willtotalsalesdiscountcalcincludeproduct</td>
<td></td>
</tr>
<tr>
<td>WILLWORKCENTERPICKINGALLOWNEGATIVEINVENTORY</td>
<td>&gt;&gt; no: False yes: True</td>
<td>msdyn_willworkcenterpickinallownegativeinventory</td>
<td></td>
</tr>
<tr>
<td>YIELDPERCENTAGE</td>
<td>&gt;</td>
<td>msdyn_yieldpercentage</td>
<td></td>
</tr>
<tr>
<td>ISUNITCOSTAUTOMATICALLYUPDATED</td>
<td>&gt;&gt; no: False yes: True</td>
<td>msdyn_isunitcostautomaticallyupdated</td>
<td></td>
</tr>
<tr>
<td>PURCHASEUNITSYMBOL</td>
<td>&gt;</td>
<td>msdyn_purchaseunitsymbol.msdyn_symbol</td>
<td></td>
</tr>
<tr>
<td>PURCHASEPRICEQUANTITY</td>
<td>&gt;</td>
<td>msdyn_purchasepricequantity</td>
<td></td>
</tr>
<tr>
<td>ISUNITCOSTINCLUDINGCHARGES</td>
<td>&gt;&gt; no: False yes: True</td>
<td>msdyn_isunitcostincludingcharges</td>
<td></td>
</tr>
<tr>
<td>FIXEDCOSTCHARGES</td>
<td>&gt;</td>
<td>msdyn_fixedcostcharges</td>
<td></td>
</tr>
<tr>
<td>MINIMUMCATCHWEIGHTQUANTITY</td>
<td>&gt;</td>
<td>msdyn_minimumcatchweightquantity</td>
<td></td>
</tr>
<tr>
<td>MAXIMUMCATCHWEIGHTQUANTITY</td>
<td>&gt;</td>
<td>msdyn_maximumcatchweightquantity</td>
<td></td>
</tr>
<tr>
<td>ALTERNATIVEITEMNUMBER</td>
<td>&gt;</td>
<td>msdyn_alternativeitemnumber.msdyn_itemnumber</td>
<td></td>
</tr>
<tr>
<td>BOMUNITSYMBOL</td>
<td>&gt;</td>
<td>msdyn_bomunitsymbol.msdyn_symbol</td>
<td></td>
</tr>
<tr>
<td>CATCHWEIGHTUNITSYMBOL</td>
<td>&gt;</td>
<td>msdyn_catchweightunitsymbol.msdyn_symbol</td>
<td></td>
</tr>
<tr>
<td>COMPARISONPRICEBASEUNITSYMBOL</td>
<td>&gt;</td>
<td>msdyn_comparisonpricebaseunitsymbol.msdyn_symbol</td>
<td></td>
</tr>
<tr>
<td>PRIMARYVENDORACCOUNTNUMBER</td>
<td>&gt;</td>
<td>msdyn_vendorid.msdyn_vendoraccountnumber</td>
<td></td>
</tr>
</tbody>
</table>
### Sales invoice headers V2 (invoices)

This template synchronizes data between Finance and Operations apps and Dataverse.

Source filter: (SalesOrderNumber != '')

Reversed source filter: msdyn_ordertype eq 192350000

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>&gt;&gt;</td>
<td>msdyn_ispricelocked</td>
<td>False</td>
</tr>
<tr>
<td>none</td>
<td>&gt;&gt;</td>
<td>statuscode</td>
<td>4</td>
</tr>
<tr>
<td>CONTACTPERSONID</td>
<td>&gt;</td>
<td>msdyn_associatedcontact.msdyn_contactforpartynumber</td>
<td></td>
</tr>
<tr>
<td>CURRENCYCODE</td>
<td>&gt;</td>
<td>transactioncurrencyid.isocurrencycode</td>
<td></td>
</tr>
<tr>
<td>CUSTOMERSORDERREFERENCE</td>
<td>&gt;</td>
<td>description</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSCITY</td>
<td>&gt;</td>
<td>billto_city</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSCOUNTRYREGIONISOCODE</td>
<td>&gt;</td>
<td>billto_country</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSSTATE</td>
<td>&gt;</td>
<td>billto_stateorprovince</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSSTREET</td>
<td>&gt;</td>
<td>billto_line1</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSSTREETNUMBER</td>
<td>&gt;</td>
<td>billto_line2</td>
<td></td>
</tr>
<tr>
<td>INVOICEADDRESSZIPCODE</td>
<td>&gt;</td>
<td>billto_postalcode</td>
<td></td>
</tr>
</tbody>
</table>

### ISCATCHWEIGHTPRODUCT

- no: False
- yes: True

### PRODUCTDIMENSIONGROUPNAME

- msdyn_productdimensiongroup.msdyn_groupname

### STORAGEDIMENSIONGROUPNAME

- msdyn_storagedimensiongroup.msdyn_groupname

### TRACKINGDIMENSIONGROUPNAME

- msdyn_trackingdimensiongroup.msdyn_groupname
## Sales invoice lines V2 (invoicedetails)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIRMEDSHIPPINGDATE</td>
<td>&gt;</td>
<td>msdyn_confirmedshippingdate</td>
<td></td>
</tr>
<tr>
<td>CURRENCYCODE</td>
<td>&gt;</td>
<td>transactioncurrencyid.isocurrencycodel</td>
<td></td>
</tr>
<tr>
<td>INVENTORYSITEID</td>
<td>&gt;</td>
<td>msdyn_inventorysite.msdyn_siteid</td>
<td></td>
</tr>
<tr>
<td>INVENTORYWAREHOUSEID</td>
<td>&gt;</td>
<td>msdyn_inventorywarehouse.msdyn_warehouseidentifier</td>
<td></td>
</tr>
<tr>
<td>INVOICEAMOUNT</td>
<td>&gt;</td>
<td>msdyn_totalamount</td>
<td></td>
</tr>
<tr>
<td>INVOICEQTY</td>
<td>&gt;</td>
<td>msdyn_quantity</td>
<td></td>
</tr>
<tr>
<td>none</td>
<td>&gt;&gt;</td>
<td>msdyn_orderid</td>
<td>1923500000</td>
</tr>
</tbody>
</table>

### Field Mapping Details

<table>
<thead>
<tr>
<th>FIELD</th>
<th>MAP TYPE</th>
<th>COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVOICECUSTOMERACCOUNTNUMBER</td>
<td>&gt;</td>
<td>customerid.Account(accounumber).Contact(msdyn_contactpersonid)</td>
<td></td>
</tr>
<tr>
<td>INVOICEDATE</td>
<td>&gt;</td>
<td>msdyn_invoicedate</td>
<td></td>
</tr>
<tr>
<td>INVOICENUMBER</td>
<td>&gt;</td>
<td>msdyn_invoicenumber</td>
<td></td>
</tr>
<tr>
<td>LEDGERVOUCHER</td>
<td>&gt;</td>
<td>msdyn_ledgervoucher</td>
<td></td>
</tr>
<tr>
<td>PAYMENTTermsNAME</td>
<td>&gt;</td>
<td>msdyn_paymentterms.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>SALESORDERNUMBER</td>
<td>&gt;</td>
<td>salesorderid.msdyn_salesordernumber</td>
<td></td>
</tr>
<tr>
<td>TOTALCHARGEAMOUNT</td>
<td>&gt;</td>
<td>freightamount</td>
<td></td>
</tr>
<tr>
<td>TOTALDISCOUNTAMOUNT</td>
<td>&gt;</td>
<td>msdyn_totaldiscountamount</td>
<td></td>
</tr>
<tr>
<td>TOTALDISCOUNTCUSTOMERGROUPCODE</td>
<td>&gt;</td>
<td>discountamount</td>
<td></td>
</tr>
<tr>
<td>TOTALINVOICEAMOUNT</td>
<td>&gt;</td>
<td>msdyn_totalamount</td>
<td></td>
</tr>
<tr>
<td>TOTALTAXAMOUNT</td>
<td>&gt;</td>
<td>msdyn_totaltax</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>INVOICENUMBER</td>
<td>&gt;</td>
<td>invoiceid.msdyn_invoicenu</td>
<td></td>
</tr>
<tr>
<td>LEDGERVOUCHER</td>
<td>&gt;</td>
<td>invoiceid.msdyn_ledgervoucher</td>
<td></td>
</tr>
<tr>
<td>LINEAMOUNT</td>
<td>&gt;</td>
<td>extendedamount</td>
<td></td>
</tr>
<tr>
<td>LINECREATIONSEQUENCENUMBER</td>
<td>&gt;</td>
<td>sequencenumber</td>
<td></td>
</tr>
<tr>
<td>LINETOTALCHARGEAMOUNT</td>
<td>&gt;</td>
<td>msdyn_totalchargeamount</td>
<td></td>
</tr>
<tr>
<td>LINETOTALDISCOUNTAMOUNT</td>
<td>&gt;</td>
<td>manualdiscountamount</td>
<td></td>
</tr>
<tr>
<td>LINETOTALTAXAMOUNT</td>
<td>&gt;</td>
<td>tax</td>
<td></td>
</tr>
<tr>
<td>PRODUCTNAME</td>
<td>&gt;</td>
<td>description</td>
<td></td>
</tr>
<tr>
<td>PRODUCTNUMBER</td>
<td>&gt;</td>
<td>productid.msdyn_productnum</td>
<td></td>
</tr>
<tr>
<td>SALESPRICE</td>
<td>&gt;</td>
<td>priceperunit</td>
<td></td>
</tr>
<tr>
<td>SALESPRODUCTCATEGORYHIERARCHYNAME</td>
<td>&gt;</td>
<td>msdyn_salesproductcategory.msdyn_hierarchy.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>SALESPRODUCTCATEGORYNAME</td>
<td>&gt;</td>
<td>msdyn_salesproductcategory.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>SALESUNITSYMBOL</td>
<td>&gt;</td>
<td>uomid.msdyn_symbol</td>
<td></td>
</tr>
<tr>
<td>none</td>
<td>&gt;&gt;</td>
<td>producttypecode</td>
<td>1</td>
</tr>
<tr>
<td>none</td>
<td>&gt;&gt;</td>
<td>propertyconfigurationstatus</td>
<td>2</td>
</tr>
<tr>
<td>none</td>
<td>&gt;&gt;</td>
<td>ispriceoverridden</td>
<td>True</td>
</tr>
</tbody>
</table>

**Sales order header document attachments (annotations)**

This template synchronizes data between Finance and Operations apps and Dataverse.

Source filter: ((DocumentAttachmentTypeCode == "Note") || (DocumentAttachmentTypeCode == "URL"))

Reversed source filter: objecttypecode eq 'salesorder'

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOCUMENTID</td>
<td>=</td>
<td>msdyn_notesid</td>
<td></td>
</tr>
<tr>
<td><strong>FINANCE AND OPERATIONS FIELD</strong></td>
<td><strong>MAP TYPE</strong></td>
<td><strong>CUSTOMER ENGAGEMENT COLUMN</strong></td>
<td><strong>DEFAULT VALUE</strong></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------</td>
<td>---------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>NOTES</td>
<td>=</td>
<td>notetext</td>
<td></td>
</tr>
<tr>
<td>ATTACHMENTDESCRIPTION</td>
<td>=</td>
<td>subject</td>
<td></td>
</tr>
<tr>
<td>SALESORDERNUMBER</td>
<td>=</td>
<td>msdyn_relatedentityid</td>
<td></td>
</tr>
<tr>
<td>DOCUMENTATTACHMENTTYPECODE</td>
<td>&lt;&lt;</td>
<td>none</td>
<td>Note</td>
</tr>
<tr>
<td>none</td>
<td>&gt;&gt;</td>
<td>objecttypecode</td>
<td>salesorder</td>
</tr>
<tr>
<td>ACCESSRESTRICTION</td>
<td>&lt;</td>
<td>msdyn_restriction</td>
<td></td>
</tr>
</tbody>
</table>

### Sales order origin codes (msdyn_salesorderorigins)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th><strong>FINANCE AND OPERATIONS FIELD</strong></th>
<th><strong>MAP TYPE</strong></th>
<th><strong>CUSTOMER ENGAGEMENT COLUMN</strong></th>
<th><strong>DEFAULT VALUE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>ORIGINCODE</td>
<td>=</td>
<td>msdyn_origincode</td>
<td></td>
</tr>
<tr>
<td>ORIGINDESCRIPTION</td>
<td>=</td>
<td>msdyn_origindescription</td>
<td></td>
</tr>
</tbody>
</table>

### Sales tax authorities (msdyn_taxauthorities)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th><strong>FINANCE AND OPERATIONS FIELD</strong></th>
<th><strong>MAP TYPE</strong></th>
<th><strong>CUSTOMER ENGAGEMENT COLUMN</strong></th>
<th><strong>DEFAULT VALUE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>TAXAUTHORITYCODE</td>
<td>=</td>
<td>msdyn_taxauthoritycode</td>
<td></td>
</tr>
<tr>
<td>TAXAUTHORITYIDENTIFICATION</td>
<td>=</td>
<td>msdyn_taxauthorityidentifier</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
</tbody>
</table>
### Sales tax exempt code entity CDS (msdyn_taxexemptcodes)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
</tbody>
</table>

### Sales tax groups (msdyn_taxgroups)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAXGROUPCODE</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
</tbody>
</table>
**Sales tax ledger posting groups V2 (msdyn_taxpostinggroups)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TaxPostingGroupCode</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
</tbody>
</table>

**Salutations (msdyn_salutations)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SalutationPhrase</td>
<td>=</td>
<td>msdyn_salutationphrase</td>
<td></td>
</tr>
</tbody>
</table>

**Sites (msdyn_operationalsites)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultFinancialDimensionValue</td>
<td>=</td>
<td>msdyn_defaultfinancialdimensionvalue</td>
<td></td>
</tr>
<tr>
<td>DefaultInventoryStatustid</td>
<td>=</td>
<td>msdyn_defaultinventorystatustid</td>
<td></td>
</tr>
<tr>
<td>IsReceivingWarehouseOverrideAllowed</td>
<td>&gt; &lt;</td>
<td>msdyn_isreceivingwarehouseoverrideallowed</td>
<td></td>
</tr>
<tr>
<td>SiteId</td>
<td>=</td>
<td>msdyn_siteid</td>
<td></td>
</tr>
<tr>
<td>SiteName</td>
<td>=</td>
<td>msdyn_sitename</td>
<td></td>
</tr>
<tr>
<td>TaxBranchCode</td>
<td>=</td>
<td>msdyn_taxbranchcode</td>
<td></td>
</tr>
<tr>
<td>FiscalEstablishmentId</td>
<td>=</td>
<td>msdyn_fiscalestablishmentid</td>
<td></td>
</tr>
<tr>
<td>IsPrimaryAddressAssigned</td>
<td>&gt; &lt;</td>
<td>msdyn_isprimaryaddressassigned</td>
<td></td>
</tr>
<tr>
<td>PrimaryAddressCity</td>
<td>=</td>
<td>msdyn_primaryaddresscity</td>
<td></td>
</tr>
<tr>
<td>PrimaryAddressCountryRegionId</td>
<td>=</td>
<td>msdyn_primaryaddresscountrylegionid</td>
<td></td>
</tr>
<tr>
<td>PrimaryAddressCountyId</td>
<td>=</td>
<td>msdyn_primaryaddresscountyid</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>----------</td>
<td>-------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>PRIMARYADDRESSDISTRICTNAME</td>
<td>=</td>
<td>msdyn_primaryaddressdistrictname</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSLATITUDE</td>
<td>=</td>
<td>msdyn_primaryaddresslatitude</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSLOCATIONROLES</td>
<td>=</td>
<td>msdyn_primaryaddresslocationrole</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSLOCATIONNSALESTAXGROUPCODE</td>
<td>=</td>
<td>msdyn_primaryaddresslocationsalestaxgroupcode</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSLONGITUDE</td>
<td>=</td>
<td>msdyn_primaryaddresslongitude</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSSTATEID</td>
<td>=</td>
<td>msdyn_primaryaddressstateid</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSSTREET</td>
<td>=</td>
<td>msdyn_primaryaddressstreet</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSZIPCODE</td>
<td>=</td>
<td>msdyn_primaryaddresszipcode</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSBUILDINGCOMPLIMENT</td>
<td>=</td>
<td>msdyn_primaryaddressbuildingcompliment</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSCITYINKANA</td>
<td>=</td>
<td>msdyn_primaryaddresscityinkana</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSSTREETINKANA</td>
<td>=</td>
<td>msdyn_primaryaddressstreetinkana</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSDESCRIPTION</td>
<td>=</td>
<td>msdyn_primaryaddressdescription</td>
<td></td>
</tr>
<tr>
<td>FORMATTEDPRIMARYADDRESS</td>
<td>=</td>
<td>msdyn_formattedprimaryaddress</td>
<td></td>
</tr>
<tr>
<td>WILLMASTERPLANNEDINTRANSAFITEMOVEMENTSUBJETRANSFERJOURNALS</td>
<td>&lt; &gt; no : False yes : True</td>
<td>msdyn_masterplannedusestransferjournal</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSPOSTBOX</td>
<td>=</td>
<td>msdyn_primaryaddresspostbox</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSSTREETNUMBER</td>
<td>=</td>
<td>msdyn_primaryaddressstreetnumber</td>
<td></td>
</tr>
</tbody>
</table>

**Sizes (msdyn_productsizes)**

This template synchronizes data between Finance and Operations apps and Dataverse.
<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIZEID</td>
<td>&gt;</td>
<td>msdyn_productsize</td>
<td></td>
</tr>
</tbody>
</table>

**Storage dimension groups (msdyn_productstoragedimensiongroups)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>WILLSALESPRICESEARCHUSEWAREHOUSE</td>
<td>&gt; &lt;</td>
<td>msdyn_willsalespricesearchusewarehouse</td>
<td></td>
</tr>
<tr>
<td>WILLSALESPRICESEARCHUSESITE</td>
<td>&gt; &lt;</td>
<td>msdyn_willsalespricesearchusesite</td>
<td></td>
</tr>
<tr>
<td>WILLSALESPRICESEARCHUSEINVENTORYSTATUS</td>
<td>&gt; &lt;</td>
<td>msdyn_willsalespricesearchuseinventorystatus</td>
<td></td>
</tr>
<tr>
<td>WILLPURCHASEPRICESEARCHUSEWAREHOUSE</td>
<td>&gt; &lt;</td>
<td>msdyn_willpurchasepricesearchusewarehouse</td>
<td></td>
</tr>
<tr>
<td>WILLPURCHASEPRICESEARCHUSESITE</td>
<td>&gt; &lt;</td>
<td>msdyn_willpurchasepricesearchusesite</td>
<td></td>
</tr>
<tr>
<td>WILLPURCHASEPRICESEARCHUSEINVENTORYSTATUS</td>
<td>&gt; &lt;</td>
<td>msdyn_willpurchasepricesearchuseinventorystatus</td>
<td></td>
</tr>
<tr>
<td>WILLCOVERAGEPLANNINGUSEWAREHOUSE</td>
<td>&gt; &lt;</td>
<td>msdyn_willcoverageplanusewarehouse</td>
<td></td>
</tr>
<tr>
<td>WILLCOVERAGEPLANNINGUSELOCATION</td>
<td>&gt; &lt;</td>
<td>msdyn_iscoverageplanenabledforlocation</td>
<td></td>
</tr>
<tr>
<td>WILLCOVERAGEPLANNINGUSEINVENTORYSTATUS</td>
<td>&gt; &lt;</td>
<td>msdyn_willcoverageplanuseinventorystatus</td>
<td></td>
</tr>
<tr>
<td>AREADVANCEDWAREHOSEMANAGEMENTPROCESSENABLED</td>
<td>&gt; &lt;</td>
<td>msdyn_areadvancedwmprocessesenabled</td>
<td></td>
</tr>
<tr>
<td>ISWAREHOUSEPRIMARYSTORAGEDIMENSION</td>
<td>&gt; &lt;</td>
<td>msdyn_iswarehouseprimarystoragedimension</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>ISWAREHOUSEMANDATORY</td>
<td>&gt;</td>
<td>msdyn_iswarehousemandatorty</td>
<td>true</td>
</tr>
<tr>
<td>ISPHYSICALINVENTORYENABLEDFORWAREHOUSE</td>
<td>&gt;</td>
<td>msdyn_isphysicalinventoryenabledforwarehouse</td>
<td>true</td>
</tr>
<tr>
<td>ISPHYSICALINVENTORYENABLEDFORLOCATION</td>
<td>&gt;</td>
<td>msdyn_isphysicalinventoryenabledforlocation</td>
<td>true</td>
</tr>
<tr>
<td>ISLOCATIONACTIVE</td>
<td>&gt;</td>
<td>msdyn_islocationactive</td>
<td>true</td>
</tr>
<tr>
<td>ISFINANCIALINVENTORYENABLEDFORWAREHOUSE</td>
<td>&gt;</td>
<td>msdyn_isfinancialinventoryenabledforwarehouse</td>
<td>true</td>
</tr>
<tr>
<td>ISBLANKRECEIPTALLOWEDFORLOCATION</td>
<td>&gt;</td>
<td>msdyn_isblankreceiptallowedforlocation</td>
<td>true</td>
</tr>
<tr>
<td>ISBLANKISSUEALLOWEDFORLOCATION</td>
<td>&gt;</td>
<td>msdyn_isblankissueallowedforlocation</td>
<td>true</td>
</tr>
</tbody>
</table>

**GROUPNAME** = msdyn_groupname

**GROUPDESCRIPTION** = msdyn_groupdescription

**FINANCE AND OPERATIONS FIELD**

**MAP TYPE**

**CUSTOMER ENGAGEMENT COLUMN**

**DEFAULT VALUE**

**Styles (msdyn_productstyles)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>STYLEID</td>
<td>&gt;</td>
<td>msdyn_productstyle</td>
<td></td>
</tr>
</tbody>
</table>

**Terms of delivery (msdyn_termsofdeliveries)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TERMSCODE</td>
<td>=</td>
<td>msdyn_termscode</td>
<td></td>
</tr>
<tr>
<td>INTRASTATCODE</td>
<td>=</td>
<td>msdyn_intrastatcode</td>
<td></td>
</tr>
</tbody>
</table>
### Finance and Operations Field

<table>
<thead>
<tr>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SaleTaxLocationRole</td>
<td>msdyn_salestaxlocationrole</td>
<td></td>
</tr>
<tr>
<td>TermsDescription</td>
<td>msdyn_termsdescription</td>
<td></td>
</tr>
<tr>
<td>FreightChargeTerms</td>
<td>msdyn_freightchargeterms</td>
<td></td>
</tr>
<tr>
<td>DoRetailSalesOrdersGeneratedTransportationChargesAdded</td>
<td>msdyn_doretailsogettransportationchargesadded</td>
<td></td>
</tr>
<tr>
<td>IsCashOnDelivery</td>
<td>msdyn_iscashondelivery</td>
<td></td>
</tr>
<tr>
<td>WillShipmentConfirmationTransferCharges</td>
<td>msdyn_willshipmentconfirmationtransfercharges</td>
<td></td>
</tr>
</tbody>
</table>

#### Terms of payment (msdyn_paymentterms)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td><strong>FINANCE AND OPERATIONS FIELD</strong></td>
<td><strong>MAP TYPE</strong></td>
<td><strong>CUSTOMER ENGAGEMENT COLUMN</strong></td>
<td><strong>DEFAULT VALUE</strong></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------</td>
<td>---------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>NUMBEROFMONTHS</td>
<td>=</td>
<td>msdyn_numberofmonth</td>
<td></td>
</tr>
<tr>
<td>CUTOFFDAYOFMONTH</td>
<td>=</td>
<td>msdyn_cutoffdayofmonth</td>
<td></td>
</tr>
<tr>
<td>ISCASHPAYMENT</td>
<td>&lt; ![no: False, yes: True]</td>
<td>msdyn_iscashpayment</td>
<td></td>
</tr>
<tr>
<td>NUMBEROFDAYS</td>
<td>=</td>
<td>msdyn_days</td>
<td></td>
</tr>
<tr>
<td>ISCERTIFIEDCOMPANYCHECK</td>
<td>&lt; ![yes: True, no: False]</td>
<td>msdyn_iscertifiedcompanycheck</td>
<td></td>
</tr>
<tr>
<td>ISDEFAULTPAYMENTTERM</td>
<td>&lt; ![no: False, yes: True]</td>
<td>msdyn_isdefaultpaymentterm</td>
<td></td>
</tr>
<tr>
<td>CREDITCARDPAYMENTTYPE</td>
<td>&lt; ![na: 806380998, creditCard: 806380999]</td>
<td>msdyn_creditcardpaymenttype</td>
<td></td>
</tr>
<tr>
<td>CREDITCARDCREDITCHECKTYPE</td>
<td>&lt; ![normal: 806380000, byPass: 806380001]</td>
<td>msdyn_creditcardcreditchecktype</td>
<td></td>
</tr>
<tr>
<td>PAYMENTDAYNAME</td>
<td>=</td>
<td>msdyn_paymentdayname.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>PAYMENTMETHODTYPE</td>
<td>&lt; ![net: 806380991, currentMth: 806380992, currentQuart: 806380993, currentYear: 806380994, currentWeek: 806380995, cod: 806380996, cutOffDate: 806380997]</td>
<td>msdyn_paymentmethodtype</td>
<td></td>
</tr>
<tr>
<td>PAYMENTSCHEDULENAME</td>
<td>=</td>
<td>msdyn_paymentscheduledname.msdyn_name</td>
<td></td>
</tr>
</tbody>
</table>

**Tracking dimension groups (msdyn_producttrackingdimensiongroups)**

This template synchronizes data between Finance and Operations apps and Dataverse.
<table>
<thead>
<tr>
<th><strong>FIELD</strong></th>
<th><strong>MAP TYPE</strong></th>
<th><strong>CUSTOMER ENGAGEMENT COLUMN</strong></th>
<th><strong>DEFAULT VALUE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>SERIALNUMBERCAPTURINGOPERATION</td>
<td>&gt;=</td>
<td>msdyn_serialnumbercapturingoperation</td>
<td></td>
</tr>
<tr>
<td>GROUPNAME</td>
<td>=</td>
<td>msdyn_groupname</td>
<td></td>
</tr>
<tr>
<td>GROUPDESCRIPTION</td>
<td>=</td>
<td>msdyn_groupdescription</td>
<td></td>
</tr>
<tr>
<td>ISSERIALNUMBERENABLEDFORPRODUCTIONCONSUMPTIONPROCESS</td>
<td>&gt;=</td>
<td>msdyn_iss-enabledforprocess</td>
<td></td>
</tr>
<tr>
<td>ISSERIALNUMBERCONTROLENABLED</td>
<td>&gt;=</td>
<td>msdyn_iss-serialnumbercontrolenabled</td>
<td></td>
</tr>
<tr>
<td>ISSERIALNUMBERENABLEDFORSALESPROCESS</td>
<td>&gt;=</td>
<td>msdyn_iss-enabledforsalesprocess</td>
<td></td>
</tr>
<tr>
<td>ISSERIALNUMBERACTIVE</td>
<td>&gt;=</td>
<td>msdyn_iss-serialnumberactive</td>
<td></td>
</tr>
<tr>
<td>ISSALESPRICEBYSERIALNUMBER</td>
<td>&gt;=</td>
<td>msdyn_iss-salespricebyserialnumber</td>
<td></td>
</tr>
<tr>
<td>ISSALESPRICEBYBATCHNUMBER</td>
<td>&gt;=</td>
<td>msdyn_iss-salespricebybatchnumber</td>
<td></td>
</tr>
<tr>
<td>ISPURCHASEPRICEBYSERIALNUMBER</td>
<td>&gt;=</td>
<td>msdyn_iss-purchasepricebyserialnumber</td>
<td></td>
</tr>
<tr>
<td>ISPURCHASEPRICEBYBATCHNUMBER</td>
<td>&gt;=</td>
<td>msdyn_iss-purchasepricebybatchnumber</td>
<td></td>
</tr>
<tr>
<td>ISPRIMARYSTOCKINGENABLEDFORSERIALNUMBER</td>
<td>&gt;=</td>
<td>msdyn_isprimarystockingenabledforserialnumber</td>
<td></td>
</tr>
<tr>
<td>ISPRIMARYSTOCKINGENABLEDFORBATCHNUMBER</td>
<td>&gt;=</td>
<td>msdyn_isprimarystockingenabledforbatchnumber</td>
<td></td>
</tr>
<tr>
<td>ISPHYSICALINVENTORYENABLEDFORSERIALNUMBER</td>
<td>&gt;=</td>
<td>msdyn_isphysicalinventoryenabledforserialnumber</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>ISPHYSICALINVENTORYENABLEDFORBATCHNUMBER</td>
<td>&gt; &lt;</td>
<td>msysn_isphysicalinventoryenabledforbn</td>
<td></td>
</tr>
<tr>
<td>ISFINANCIALINVENTORYENABLEDFORSERIALNUMBER</td>
<td>&gt; &lt;</td>
<td>msysn_isfinancialinventoryenabledforsn</td>
<td></td>
</tr>
<tr>
<td>ISFINANCIALINVENTORYENABLEDFORBATCHNUMBER</td>
<td>&gt; &lt;</td>
<td>msysn_isfinancialinventoryenabledforbn</td>
<td></td>
</tr>
<tr>
<td>ISCOVERAGENPLANENABLEDFORSERIALNUMBER</td>
<td>&gt; &lt;</td>
<td>msysn_iscoverageplanenableforallserialnumber</td>
<td></td>
</tr>
<tr>
<td>ISCOVERAGENPLANENABLEDFORBATCHNUMBER</td>
<td>&gt; &lt;</td>
<td>msysn_iscoverageplanenableforbatchnumber</td>
<td></td>
</tr>
<tr>
<td>ISBLANKRECEIPTALLOWEDFORSERIALNUMBER</td>
<td>&gt; &lt;</td>
<td>msysn_isblankreceiptallowedforallserialnumber</td>
<td></td>
</tr>
<tr>
<td>ISBLANKRECEIPTALLOWEDFORBATCHNUMBER</td>
<td>&gt; &lt;</td>
<td>msysn_isblankreceiptallowedforbatchnumber</td>
<td></td>
</tr>
<tr>
<td>ISBLANKISSUEALLOWEDFORSERIALNUMBER</td>
<td>&gt; &lt;</td>
<td>msysn_isblankissuelowedforallserialnumber</td>
<td></td>
</tr>
<tr>
<td>ISBLANKISSUEALLOWEDFORBATCHNUMBER</td>
<td>&gt; &lt;</td>
<td>msysn_isblankissuelowedforbatchnumber</td>
<td></td>
</tr>
<tr>
<td>ISBATCHNUMBERACTIVE</td>
<td>&gt; &lt;</td>
<td>msysn_isbatchnumberactive</td>
<td></td>
</tr>
<tr>
<td>ISINVENTORYOWNERACTIVE</td>
<td>&gt; &lt;</td>
<td>msysn_isinventoryowneractive</td>
<td></td>
</tr>
</tbody>
</table>

**Unit conversions (msdyn_unitofmeasureconversions)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DENOMINATOR</td>
<td>=</td>
<td>msysn_denominator</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------</td>
<td>-----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>NUMERATOR</td>
<td>=</td>
<td>msdyn_numerator</td>
<td></td>
</tr>
<tr>
<td>FACTOR</td>
<td>=</td>
<td>msdyn_factor</td>
<td></td>
</tr>
<tr>
<td>INNEROFFSET</td>
<td>=</td>
<td>msdyn_inneroffset</td>
<td></td>
</tr>
<tr>
<td>OUTEROFFSET</td>
<td>=</td>
<td>msdyn_outeroffset</td>
<td></td>
</tr>
<tr>
<td>ROUNDDING</td>
<td>&gt; &lt;</td>
<td>msdyn_rounding</td>
<td></td>
</tr>
<tr>
<td>TOUNITSYMBOL</td>
<td>=</td>
<td>msdyn_tounit.msdyn_symbol</td>
<td></td>
</tr>
<tr>
<td>FROMUNITSYMBOL</td>
<td>=</td>
<td>msdyn_fromunit.msdyn_symbol</td>
<td></td>
</tr>
</tbody>
</table>

**Units (uoms)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNITSYMBOL</td>
<td>&gt;</td>
<td>msdyn_symbol</td>
<td></td>
</tr>
<tr>
<td>UNITCLASS</td>
<td>&gt;</td>
<td>msdyn_externalunitclassname</td>
<td></td>
</tr>
<tr>
<td>DECIMALPRECISION</td>
<td>&gt;</td>
<td>msdyn_decimalprecision</td>
<td></td>
</tr>
<tr>
<td>ISBASEUNIT</td>
<td>&gt;&gt;</td>
<td>msdyn_isbaseunit</td>
<td></td>
</tr>
<tr>
<td>ISSYSTEMUNIT</td>
<td>&gt;&gt;</td>
<td>msdyn_issystemunit</td>
<td></td>
</tr>
<tr>
<td>SYSTEMOFUNITS</td>
<td>&gt;&gt;</td>
<td>msdyn_systemofunits</td>
<td></td>
</tr>
<tr>
<td>UNITSYMBOL</td>
<td>&gt;</td>
<td>name</td>
<td></td>
</tr>
<tr>
<td>UNITDESCRIPTION</td>
<td>&gt;</td>
<td>msdyn_description</td>
<td></td>
</tr>
</tbody>
</table>

**Vendor document attachments (annotations)**

This template synchronizes data between Finance and Operations apps and Dataverse.
Source filter: ((DocumentAttachmentTypeCode == "Note") || (DocumentAttachmentTypeCode == "URL"))

Reversed source filter: (objecttypecode eq 'msdyn_vendor' and msdyn_relatedentityid2 ne null) or (objecttypecode eq 'account' and msdyn_relatedentityid2 ne null) or (objecttypecode eq 'contact' and msdyn_relatedentityid2 ne null)

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOCUMENTID</td>
<td>=</td>
<td>msdyn_notesid</td>
<td></td>
</tr>
<tr>
<td>NOTES</td>
<td>=</td>
<td>notetext</td>
<td></td>
</tr>
<tr>
<td>ATTACHMENTDESCRIPTION</td>
<td>=</td>
<td>subject</td>
<td></td>
</tr>
<tr>
<td>VENDORACCOUNTNUMBER</td>
<td>=</td>
<td>msdyn_relatedentityid2</td>
<td></td>
</tr>
<tr>
<td>DOCUMENTATTACHMENTTYPECODE</td>
<td>&lt;&lt;</td>
<td>none</td>
<td>Note</td>
</tr>
<tr>
<td>none</td>
<td>&gt;&gt;</td>
<td>objecttypecode</td>
<td>msdyn_vendor</td>
</tr>
<tr>
<td>ACCESSRESTRICTION</td>
<td>&gt; &lt;</td>
<td>msdyn_restriction</td>
<td></td>
</tr>
<tr>
<td></td>
<td>internal : 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>external : 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Vendor groups (msdyn_vendorgroups)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULTPAYMENTTERMMENAME</td>
<td>=</td>
<td>msdyn_paymentterms.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
<tr>
<td>VENDORGROUPID</td>
<td>=</td>
<td>msdyn_vendorgroup</td>
<td></td>
</tr>
<tr>
<td>CLEARINGPERIODPAYMENTTERMNAME</td>
<td>=</td>
<td>msdyn_clearingperiodpaymenttermname.msdyn_name</td>
<td></td>
</tr>
</tbody>
</table>

Vendor payment method (msdyn_vendorpaymentmethods)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
<tr>
<td><strong>FINANCE AND OPERATIONS FIELD</strong></td>
<td><strong>MAP TYPE</strong></td>
<td><strong>CUSTOMER ENGAGEMENT COLUMN</strong></td>
<td><strong>DEFAULT VALUE</strong></td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------</td>
<td>---------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td><code>SUMBYPERIOD</code></td>
<td><code>&gt;</code></td>
<td><code>msdyn_sumbyperiod</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>&lt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>transDate</code>: 806380001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>invoice</code>: 806380000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>week</code>: 806380002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>total</code>: 806380003</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>DISCOUNTGRACEPERIODDAYS</code></td>
<td><code>=</code></td>
<td><code>msdyn_discountgraceperioddays</code></td>
<td></td>
</tr>
<tr>
<td><code>PAYMENTSTATUS</code></td>
<td><code>&gt;</code></td>
<td><code>msdyn_paymentstatus</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>&lt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>none</code>: 806380000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>sent</code>: 806380001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>recieved</code>: 806380002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>approved</code>: 806380003</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>rejected</code>: 806380004</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>ALLOWPAYMENTCOPIES</code></td>
<td><code>&gt;</code></td>
<td><code>msdyn_allowpaymentcopies</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>&lt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>no</code>: False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>yes</code>: True</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>PAYMENTTYPE</code></td>
<td><code>&gt;</code></td>
<td><code>msdyn_paymenttype</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>&lt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>blank</code>: 806380000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>check</code>: 806380001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>electronicPayment</code>: 806380002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>promissoryNote</code>: 806380003</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>LASTFILENUMBER</code></td>
<td><code>=</code></td>
<td><code>msdyn_lastfilenumber</code></td>
<td></td>
</tr>
<tr>
<td><code>LASTFILENUMBERTODAY</code></td>
<td><code>=</code></td>
<td><code>msdyn_lastfilenumbertoday</code></td>
<td></td>
</tr>
<tr>
<td><code>ACCOUNTTYPE</code></td>
<td><code>&gt;</code></td>
<td><code>msdyn_accounttype</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>&lt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>ledger</code>: 806380000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>bank</code>: 806380005</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>cust</code>: 806380001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>fixedAssets</code>: 806380004</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>vend</code>: 806380002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>project</code>: 806380003</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>BRIDGINGPOSTINGENABLED</code></td>
<td><code>&gt;</code></td>
<td><code>msdyn_bridgingposting</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>&lt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>no</code>: False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>yes</code>: True</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>ENABLEPOSTDATEDCHECKCLEARINGPOSTING</code></td>
<td><code>&gt;</code></td>
<td><code>msdyn_postdatedcheckclearingposting</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>&lt;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>no</code>: False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>yes</code>: True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>PROMISSORYNOTEDRAFTTYPE</td>
<td></td>
<td>msdyn_promissorynotedrafttype</td>
<td></td>
</tr>
<tr>
<td></td>
<td>noDraft: 806380000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>noAcceptance: 806380001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>acceptance: 806380002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>promissory: 806380003</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bankAcceptance: 806380004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIRECTDEBIT</td>
<td></td>
<td>msdyn_directdebit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no: False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes: True</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Vendors V2 (msdyn_vendors)**

This template synchronizes data between Finance and Operations apps and Dataverse.

Source filter: (VendorPartyType == "Organization") || (VendorPartyType == "Person")

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>VENDORACCOUNTNUMBER</td>
<td></td>
<td>msdyn_vendoraccountnumber</td>
<td></td>
</tr>
<tr>
<td>VENDORGROUPID</td>
<td></td>
<td>msdyn_vendorgroupid.msdyn_vendorgroup</td>
<td></td>
</tr>
<tr>
<td>VENDORPARTYTYPE</td>
<td></td>
<td>msdyn_isperson</td>
<td></td>
</tr>
<tr>
<td></td>
<td>person: True</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>organization: False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CREDITLIMIT</td>
<td></td>
<td>msdyn_vendorcreditlimit</td>
<td></td>
</tr>
<tr>
<td>ISFOREIGNENTITY</td>
<td></td>
<td>msdyn_isforeignentity</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no: False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes: True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISONETIMEVENDOR</td>
<td></td>
<td>msdyn_isonetimevendor</td>
<td></td>
</tr>
<tr>
<td></td>
<td>no: False</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes: True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CREDITRATING</td>
<td></td>
<td>msdyn_creditrating</td>
<td></td>
</tr>
<tr>
<td>DUNSNUMBER</td>
<td></td>
<td>msdyn_dunsnnumber</td>
<td></td>
</tr>
<tr>
<td>ETHNICORIGINID</td>
<td></td>
<td>msdyn_ethnicorigin</td>
<td></td>
</tr>
<tr>
<td>OURACCOUNTNUMBER</td>
<td></td>
<td>msdyn_ourvendoraccountnumber</td>
<td></td>
</tr>
<tr>
<td>PAYMENTID</td>
<td></td>
<td>msdyn_paymentid</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT FIELD</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------</td>
<td>---------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>PRIMARYURL</td>
<td>=</td>
<td>msdyn_primarycontacturl</td>
<td></td>
</tr>
<tr>
<td>DEFAULTPAYMENTDAYNAME</td>
<td>=</td>
<td>msdyn_defaultpaymentdayname.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>DEFAULTPAYMENTSCHEDULENAME</td>
<td>=</td>
<td>msdyn_paymentschedule.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>DEFAULTPAYMENTTERMSNAMEME</td>
<td>=</td>
<td>msdyn_paymentterms.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>HASONLYTAKENBIDS</td>
<td>&lt;</td>
<td>msdyn_hasonlytakenbids</td>
<td></td>
</tr>
<tr>
<td>ISMINORITYOWNED</td>
<td>&lt;</td>
<td>msdyn_isminorityowned</td>
<td></td>
</tr>
<tr>
<td>ISVENDORLOCALLYOWNED</td>
<td>&lt;</td>
<td>msdyn_isvendorlocallyowned</td>
<td></td>
</tr>
<tr>
<td>ISSERVICEVETERANOWNED</td>
<td>&lt;</td>
<td>msdyn_isserviceveteranowned</td>
<td></td>
</tr>
<tr>
<td>ISOWNERDISABLED</td>
<td>&lt;</td>
<td>msdyn_ownerisdisabled</td>
<td></td>
</tr>
<tr>
<td>ISWOMANOWNER</td>
<td>&lt;</td>
<td>msdyn_womanowner</td>
<td></td>
</tr>
<tr>
<td>ORGANIZATIONEMPLOYEEAMOUNT</td>
<td>=</td>
<td>msdyn_numberofemployees</td>
<td></td>
</tr>
<tr>
<td>VENDORHOLDRELEASEDATE</td>
<td>=</td>
<td>msdyn_vendoronholdreleasedate</td>
<td></td>
</tr>
<tr>
<td>VENDORPARTYNUMBER</td>
<td>=</td>
<td>msdyn_partyid.msdyn_partynumber</td>
<td></td>
</tr>
<tr>
<td>ONHOLDSTATUS</td>
<td>&lt;</td>
<td>msdyn_onholdstatus</td>
<td></td>
</tr>
</tbody>
</table>

```python
no : 806380000
invoice : 806380001
all : 806380002
payment : 806380003
requisition : 806380004
never : 806380005
```
<table>
<thead>
<tr>
<th><strong>FINANCE AND OPERATIONS FIELD</strong></th>
<th><strong>MAP TYPE</strong></th>
<th><strong>CUSTOMER ENGAGEMENT COLUMN</strong></th>
<th><strong>DEFAULT VALUE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>CURRENCYCODE</td>
<td>=</td>
<td>msdyn_currencycode.isocurrencycode</td>
<td></td>
</tr>
<tr>
<td>ISVENDORLOCATEDINHUBZONE</td>
<td>&gt; &lt;</td>
<td>msdyn_isvendorlocatedinhubzone</td>
<td></td>
</tr>
<tr>
<td>DEFAULTVENDORPAYMENTMETHODNAME</td>
<td>=</td>
<td>msdyn_vendorpaymentmethod.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>INVOICEVENDORACCOUNTNUMBER</td>
<td>=</td>
<td>msdyn_invoicevendoraccountnumber.msdyn_vendoraccountnumber</td>
<td></td>
</tr>
<tr>
<td>AREPRICESINCLUDINGSALESTAX</td>
<td>&gt; &lt;</td>
<td>msdyn_priceincludessalestax</td>
<td></td>
</tr>
<tr>
<td>SALESTAXGROUPCODE</td>
<td>=</td>
<td>msdyn_taxgroup.msdyn_name</td>
<td></td>
</tr>
<tr>
<td>PRIMARYCONTACTPERSONID</td>
<td>=</td>
<td>msdyn_primarycontact.msdyn_contactforpartynumber</td>
<td></td>
</tr>
</tbody>
</table>

**Veteran status (cdm_veteranstatuses)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th><strong>FINANCE AND OPERATIONS FIELD</strong></th>
<th><strong>MAP TYPE</strong></th>
<th><strong>CUSTOMER ENGAGEMENT COLUMN</strong></th>
<th><strong>DEFAULT VALUE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>VETERANSTATUSID</td>
<td>=</td>
<td>cdm_name</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>=</td>
<td>cdm_description</td>
<td></td>
</tr>
<tr>
<td>ISPROTECTEDVETERAN</td>
<td>&gt; &lt;</td>
<td>cdm_isprotectedveteran</td>
<td></td>
</tr>
</tbody>
</table>

**Warehouse locations (msdyn_inventorylocations)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th><strong>FINANCE AND OPERATIONS FIELD</strong></th>
<th><strong>MAP TYPE</strong></th>
<th><strong>CUSTOMER ENGAGEMENT COLUMN</strong></th>
<th><strong>DEFAULT VALUE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>BINID</td>
<td>=</td>
<td>msdyn_binid</td>
<td></td>
</tr>
<tr>
<td>CHECKDIGITS</td>
<td>=</td>
<td>msdyn_checkdigits</td>
<td></td>
</tr>
<tr>
<td>GENERATECHECKDIGITS</td>
<td>&gt; &lt;</td>
<td>msdyn_generatecheckdigits</td>
<td></td>
</tr>
<tr>
<td>Finance and Operations Field</td>
<td>Map Type</td>
<td>Customer Engagement Field</td>
<td>Default Value</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>---------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Issortordercodemanual</td>
<td>&lt;</td>
<td>msdyn_issortordercodemanual</td>
<td></td>
</tr>
<tr>
<td>Iswarehouselocationid</td>
<td>&lt;</td>
<td>msdyn_iswarehouselocationidmanu</td>
<td></td>
</tr>
<tr>
<td>Lastcountedutcdateteme</td>
<td>=</td>
<td>msdyn_lastcountedutcdateteme</td>
<td></td>
</tr>
<tr>
<td>Physicaldepth</td>
<td>=</td>
<td>msdyn_physicaldepth</td>
<td></td>
</tr>
<tr>
<td>Physicalheight</td>
<td>=</td>
<td>msdyn_physicalheight</td>
<td></td>
</tr>
<tr>
<td>Physicalheightaboveground</td>
<td>=</td>
<td>msdyn_physicalheightaboveground</td>
<td></td>
</tr>
<tr>
<td>Physicalmaximumstorgevolume</td>
<td>=</td>
<td>msdyn_physicalmaximumstorgevolume</td>
<td></td>
</tr>
<tr>
<td>Physicalmaximumstorgeweight</td>
<td>=</td>
<td>msdyn_physicalmaximumstorgeweight</td>
<td></td>
</tr>
<tr>
<td>Physicalwidth</td>
<td>=</td>
<td>msdyn_physicalwidth</td>
<td></td>
</tr>
<tr>
<td>Rackid</td>
<td>=</td>
<td>msdyn_rackid</td>
<td></td>
</tr>
<tr>
<td>Shelfid</td>
<td>=</td>
<td>msdyn_shelfid</td>
<td></td>
</tr>
<tr>
<td>Warehouseid</td>
<td>=</td>
<td>msdyn_warehouse.msdyn_warehouseidentifier</td>
<td></td>
</tr>
<tr>
<td>Warehouselocationid</td>
<td>=</td>
<td>msdyn_warehouselocationid</td>
<td></td>
</tr>
<tr>
<td>Warehouselocationprorfileid</td>
<td>=</td>
<td>msdyn_warehouselocationprorfileid</td>
<td></td>
</tr>
<tr>
<td>Warehouselocationtype</td>
<td>&lt;</td>
<td>msdyn_warehouselocationtype</td>
<td></td>
</tr>
<tr>
<td>Inputwarehouselocationblockingcauseid</td>
<td>=</td>
<td>msdyn_inputwarehouselocationblockingcauseid</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Map Type Examples:**

- **Buffer:** 192350000
- **Pick:** 192350001
- **InputPort:** 192350002
- **OutputPort:** 192350003
- **InspectionLocation:** 192350004
- **KanbanSupermarket:** 192350005
<table>
<thead>
<tr>
<th>FIELD</th>
<th>MAP TYPE</th>
<th>COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUTWAREHOUSELOCATIONBLOCKINGCAUSEID</td>
<td>=</td>
<td>msdyn_outputwarehouselocationblockingcauseid</td>
<td></td>
</tr>
<tr>
<td>ISDEFAULTCREDITONLYRETURNSWAREHOUSELOCATION</td>
<td>&lt;</td>
<td>msdyn_isdefaultcreditonlyreturnwarehouseloc</td>
<td></td>
</tr>
<tr>
<td>ISDEFAULTISSUEWAREHOUSELOCATION</td>
<td>&lt;</td>
<td>msdyn_isdefaultissuewarehouselocation</td>
<td></td>
</tr>
<tr>
<td>ISDEFAULTKANBANFINISHEDGOODSWAREHOUSELOCATION</td>
<td>&lt;</td>
<td>msdyn_isdefaultkanbanfinishedgoodswarehouseloc</td>
<td></td>
</tr>
<tr>
<td>ISDEFAULTPRODUCTIONFINISHEDGOODSWAREHOUSELOCATION</td>
<td>&lt;</td>
<td>msdyn_isdefaultproductionfinishedgoodswarehouseloc</td>
<td></td>
</tr>
<tr>
<td>ISDEFAULTPRODUCTIONINPUTWAREHOUSELOCATION</td>
<td>&lt;</td>
<td>msdyn_isdefaultproductioninputwarehouseloc</td>
<td></td>
</tr>
<tr>
<td>ISDEFAULTRECEIPTWAREHOUSELOCATION</td>
<td>&lt;</td>
<td>msdyn_isdefaultreceiptwarehouselocation</td>
<td></td>
</tr>
<tr>
<td>ISDEFAULTRETAILSTORERETURNWAREHOUSELOCATION</td>
<td>&lt;</td>
<td>msdyn_isdefaultretailstorereturnwarehouseloc</td>
<td></td>
</tr>
<tr>
<td>ISDEFAULTRETAILSTOREWAREHOUSELOCATION</td>
<td>&lt;</td>
<td>msdyn_isdefaultretailstorewarehouselocation</td>
<td></td>
</tr>
<tr>
<td>ISDEFAULTSYSTEMMAINTENANCEWAREHOUSELOCATION</td>
<td>&lt;</td>
<td>msdyn_isdefaultsystemmaintenancewarehouselocation</td>
<td></td>
</tr>
<tr>
<td>AGINGDATE</td>
<td>&gt;</td>
<td>msdyn_agingdate</td>
<td></td>
</tr>
<tr>
<td>LASTACTIVITYDATETIME</td>
<td>&gt;</td>
<td>msdyn_lastactivitydatetime</td>
<td></td>
</tr>
<tr>
<td>LOCATIONSTATUS</td>
<td>&gt;</td>
<td>msdyn_locationstatus</td>
<td></td>
</tr>
<tr>
<td></td>
<td>undetermined : 192350000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>empty : 192350001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>picking : 192350002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>storage : 192350003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>-----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>ISITEMINLOCATIONMAINTAINED</td>
<td>&gt;=</td>
<td>msdyn_isiteminlocationmaintained</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td>False</td>
</tr>
<tr>
<td>ISLOCATIONACTIVITYDATETIMEMAINTAINED</td>
<td>&gt;=</td>
<td>msdyn_islocationactivitydatetimemaintained</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td>False</td>
</tr>
<tr>
<td>ISLOCATIONSTATUSMAINTAINED</td>
<td>&gt;=</td>
<td>msdyn_islocationstatusmaintained</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td>False</td>
</tr>
<tr>
<td>ISDEFAULTQUALITYMAINTENANCEWAREHOUSELOCATION</td>
<td>&gt;=</td>
<td>msdyn_isdefaultqualitymaintenancewarehouselocation</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td>False</td>
</tr>
<tr>
<td>WAREHOUSEAISLEID</td>
<td>=</td>
<td>msdyn_warehouseaisle.msdyn_aisleid</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEID</td>
<td>&gt;</td>
<td>msdyn_warehouseaisle.msdyn_warehouse.msdyn_warehouseidentifier</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEZONEID</td>
<td>=</td>
<td>msdyn_warehousezone.msdyn_warehouseidentifier</td>
<td></td>
</tr>
<tr>
<td>FIRSTADDITIONALWAREHOUSEZONEID</td>
<td>=</td>
<td>msdyn_firstadditionalwarehousezone.msdyn_zoneid</td>
<td></td>
</tr>
<tr>
<td>SECONDADDITIONALWAREHOUSEZONEID</td>
<td>=</td>
<td>msdyn_secondadditionalwarehousezone.msdyn_zoneid</td>
<td></td>
</tr>
<tr>
<td>THIRDADDITIONALWAREHOUSEZONEID</td>
<td>=</td>
<td>msdyn_thirdadditionalwarehousezone.msdyn_zoneid</td>
<td></td>
</tr>
<tr>
<td>ITEMNUMBERINLOCATION</td>
<td>=</td>
<td>msdyn_itemnumberinlocation.msdyn_itemnumber</td>
<td></td>
</tr>
</tbody>
</table>

Warehouse work headers (msdyn_warehouseworkheaders)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTUALPROCESSINGTIMESTECONDS</td>
<td>&gt;</td>
<td>msdyn_actualprocessingtime</td>
<td>seconds</td>
</tr>
<tr>
<td>ESTIMATEDPROCESSINGTIMESECONDS</td>
<td>&gt;</td>
<td>msdyn_estimatedprocessing</td>
<td>time</td>
</tr>
<tr>
<td><strong>FINANCE AND OPERATIONS FIELD</strong></td>
<td><strong>MAP TYPE</strong></td>
<td><strong>CUSTOMER ENGAGEMENT COLUMN</strong></td>
<td><strong>DEFAULT VALUE</strong></td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------</td>
<td>-------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>ISWAREHOUSEWORKBLOCKED</td>
<td>&gt;&gt;</td>
<td>msdyn_iswarehouseworkblocked</td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEID</td>
<td>&gt;</td>
<td>msdyn_warehouse.msdyn_warehous eidentifier</td>
<td></td>
</tr>
<tr>
<td>INVENTORYSITEID</td>
<td>&gt;</td>
<td>msdyn_inventorysite.msdyn_sitel id</td>
<td></td>
</tr>
<tr>
<td>ISWAREHOUSEWORKERMANUALLYASSIGNED</td>
<td>&gt;&gt;</td>
<td>msdyn_iswarehouseworkermanuallyassigned</td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : True</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEWORKCANCELLEDATETIME</td>
<td>&gt;</td>
<td>msdyn_warehouseworkcancelledat etime</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEWORKCLOSEDATETIME</td>
<td>&gt;</td>
<td>msdyn_warehouseworkclosedatet ime</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEWORKID</td>
<td>&gt;</td>
<td>msdyn_warehouseworkid</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEWORKPROCESSINGSTARTDATETIME</td>
<td>&gt;</td>
<td>msdyn_warehouseworkproc essingstartdatetime</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEWORKPRIORITY</td>
<td>&gt;</td>
<td>msdyn_warehouseworkpriority</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEWORKSTATUS</td>
<td>&gt;&gt;</td>
<td>msdyn_warehouseworkstatus</td>
<td></td>
</tr>
<tr>
<td></td>
<td>open : 192350000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>inprocess : 192350001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>skipped : 192350002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>closed : 192350003</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>cancelled : 192350004</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>combined : 192350005</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FIELD</strong></td>
<td><strong>MAP TYPE</strong></td>
<td><strong>CUSTOMER ENGAGEMENT COLUMN</strong></td>
<td><strong>DEFAULT VALUE</strong></td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------</td>
<td>---------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>CONTAINERID</td>
<td>&gt;</td>
<td>msdyn_containerid</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEWORKLOCKING</td>
<td>&gt;&gt;</td>
<td>msdyn_warehouseworklockingu</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEMOBILEDEVICE</td>
<td></td>
<td>ngmobiledeviceuserid</td>
<td></td>
</tr>
<tr>
<td>TARGETLICENSEPLATENumber</td>
<td>&gt;&gt;</td>
<td>msdyn_targetlicenseplatenu</td>
<td></td>
</tr>
<tr>
<td>WAVEID</td>
<td>&gt;</td>
<td>msdyn_waveid</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEWORKCANCELING</td>
<td>&gt;&gt;</td>
<td>msdyn_warehouseworkcancellingu</td>
<td></td>
</tr>
<tr>
<td>USERID</td>
<td></td>
<td>erid</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEWORKMANUALLY</td>
<td>&gt;&gt;</td>
<td>msdyn_warehouseworkmanuallyco</td>
<td></td>
</tr>
<tr>
<td>COMPLETINGUSERID</td>
<td></td>
<td>mpltycompletinguserid</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>WAREHOUSEWORKPOOLID</td>
<td>&gt;</td>
<td>msdyn_warehouseworkpoolid</td>
<td></td>
</tr>
</tbody>
</table>

### Warehouse work lines (msdyn_warehouseworklines)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS FIELD</th>
<th>MAP TYPE</th>
<th>CUSTOMER ENGAGEMENT COLUMN</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTUALPROCESSINGTIMESTECONDS</td>
<td>&gt;</td>
<td>msdyn_actualprocessingtimeseconds</td>
<td></td>
</tr>
<tr>
<td>ESTIMATEDPROCESSINGTIMESTECONDS</td>
<td>&gt;</td>
<td>msdyn_estimatedprocessingtimeseconds</td>
<td></td>
</tr>
<tr>
<td>FEFOTEMBEATCHNUMBER</td>
<td>&gt;</td>
<td>msdyn_fefotembeatchnumber</td>
<td></td>
</tr>
<tr>
<td>REMAININGHANDLINGQUANTITY</td>
<td>&gt;</td>
<td>msdyn_remaininghandlingquantity</td>
<td></td>
</tr>
<tr>
<td>HANDLINGQUANTITY</td>
<td>&gt;</td>
<td>msdyn_handlingquantity</td>
<td></td>
</tr>
<tr>
<td>WORKLINENUMBER</td>
<td>&gt;</td>
<td>msdyn_worklinenumber</td>
<td></td>
</tr>
<tr>
<td>ISWORKLINEMANDATORY</td>
<td>&gt;&gt;</td>
<td>msdyn_isworklinemandatory</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>yes</strong></td>
<td>True</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>no</strong></td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>REMAININGWORKQUANTITY</td>
<td>&gt;</td>
<td>msdyn_remainingworkquantity</td>
<td></td>
</tr>
<tr>
<td>WORKQUANTITY</td>
<td>&gt;</td>
<td>msdyn_workquantity</td>
<td></td>
</tr>
<tr>
<td>ISREPLENISHMENTNEEDED</td>
<td>&gt;&gt;</td>
<td>msdyn_isreplenishmentneeded</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>yes</strong></td>
<td>True</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>no</strong></td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>SORTORDERCODE</td>
<td>&gt;</td>
<td>msdyn_sortordercode</td>
<td></td>
</tr>
<tr>
<td>WORKQUANTITYUNITSYMBOL</td>
<td>&gt;</td>
<td>msdyn_workquantityunitsymbol</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEWORKCLOSEDDATETIME</td>
<td>&gt;</td>
<td>msdyn_warehouseworkcloseddatetime</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEWORKID</td>
<td>&gt;</td>
<td>msdyn_warehouseworkid</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEWORKPROCESSINGSTARTDATETIME</td>
<td>&gt;</td>
<td>msdyn_warehouseworkprocessingstartdatetime</td>
<td></td>
</tr>
<tr>
<td><strong>FIELD</strong></td>
<td><strong>MAP TYPE</strong></td>
<td><strong>CUSTOMER ENGAGEMENT COLUMN</strong></td>
<td><strong>DEFAULT VALUE</strong></td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>--------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>ISWORKEXECUTIONSTOPPED</td>
<td>yes: True, no: False</td>
<td>msdyn_isworkexecutionstopped</td>
<td></td>
</tr>
<tr>
<td>EXTRAHANDLINGQUANTITY</td>
<td>&gt;</td>
<td>msdyn_extrahandlingquantity</td>
<td></td>
</tr>
<tr>
<td>CAPTUREDWEIGHT</td>
<td>&gt;</td>
<td>msdyn_capturedweight</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEID</td>
<td>&gt;</td>
<td>msdyn_warehouse.msdyn_warehouseidentifier</td>
<td></td>
</tr>
<tr>
<td>INVENTORYSITEID</td>
<td>&gt;</td>
<td>msdyn_inventorysite.msdyn_siteid</td>
<td></td>
</tr>
<tr>
<td>CONTAINERID</td>
<td>&gt;</td>
<td>msdyn_containerid</td>
<td></td>
</tr>
<tr>
<td>PROCESSINGWAREHOUSEMOBILEDEVICEUSERID</td>
<td>&gt;</td>
<td>msdyn_processingwarehouseemobiledeviceuserid</td>
<td></td>
</tr>
<tr>
<td>ITEMBATCHNUMBER</td>
<td>&gt;</td>
<td>msdyn ITEMBATCHNUMBER</td>
<td></td>
</tr>
<tr>
<td>INVENTORYOWNERID</td>
<td>&gt;</td>
<td>msdyn_INVENTORYOWNERID</td>
<td></td>
</tr>
<tr>
<td>ITEMSERIALNUMBER</td>
<td>&gt;</td>
<td>msdyn ITEMSERIALNUMBER</td>
<td></td>
</tr>
<tr>
<td><strong>Finance and Operations Field</strong></td>
<td><strong>Map Type</strong></td>
<td><strong>Customer Engagement Column</strong></td>
<td><strong>Default Value</strong></td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------</td>
<td>-------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>INVENTORYSTATUSID</td>
<td>&gt;</td>
<td>msdyn_inventorystatusid</td>
<td></td>
</tr>
<tr>
<td>LICENSEPLATENUMBER</td>
<td>&gt;</td>
<td>msdyn_licenseplatenumber</td>
<td></td>
</tr>
<tr>
<td>ITEMNUMBER</td>
<td>&gt;</td>
<td>msdyn_item.msdyn_itemnumber</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEZONEID</td>
<td>&gt;</td>
<td>msdyn_warehousezone.msdyn_zoneid</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSELOCATIONID</td>
<td>&gt;</td>
<td>msdyn_warehouselocation.msdyn_warehouselocationid</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEID</td>
<td>&gt;</td>
<td>msdyn_warehouselocation.msdyn_warehouseidentifier</td>
<td></td>
</tr>
<tr>
<td>PRODUCTCONFIGURATIONID</td>
<td>&gt;</td>
<td>msdyn_productconfiguration.msdyn_productconfiguration</td>
<td></td>
</tr>
<tr>
<td>PRODUCTCOLORID</td>
<td>&gt;</td>
<td>msdyn_productcolor.msdyn_productcolorname</td>
<td></td>
</tr>
<tr>
<td>PRODUCTSIZEID</td>
<td>&gt;</td>
<td>msdyn_productsize.msdyn_productsize</td>
<td></td>
</tr>
<tr>
<td>PRODUCTSTYLEID</td>
<td>&gt;</td>
<td>msdyn_productstyle.msdyn_productstyle</td>
<td></td>
</tr>
</tbody>
</table>

**Warehouse zone groups (msdyn_warehousezonegroups)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th><strong>Finance and Operations Field</strong></th>
<th><strong>Map Type</strong></th>
<th><strong>Customer Engagement Column</strong></th>
<th><strong>Default Value</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUPID</td>
<td>=</td>
<td>msdyn_groupid</td>
<td></td>
</tr>
<tr>
<td>GROUPNAME</td>
<td>=</td>
<td>msdyn_groupname</td>
<td></td>
</tr>
</tbody>
</table>

**Warehouse zones (msdyn_warehousezones)**

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th><strong>Finance and Operations Field</strong></th>
<th><strong>Map Type</strong></th>
<th><strong>Customer Engagement Column</strong></th>
<th><strong>Default Value</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>WAREHOUSEZONEGROUPID</td>
<td>=</td>
<td>msdyn_warehousezonegroup.msdyn_groupid</td>
<td></td>
</tr>
<tr>
<td>ZONEID</td>
<td>=</td>
<td>msdyn_zoneid</td>
<td></td>
</tr>
<tr>
<td>ZONENAME</td>
<td>=</td>
<td>msdyn_zonename</td>
<td></td>
</tr>
</tbody>
</table>
# Warehouses (msdyn_warehouses)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULTCONTAINERTYPEID</td>
<td>=</td>
<td>msdyn_defaultcontainertyp eid</td>
<td></td>
</tr>
<tr>
<td>AREITEMSCOVERAGEPLAN</td>
<td>&gt; &lt;</td>
<td>msdyn_areitemscoverageplan edmanually</td>
<td></td>
</tr>
<tr>
<td>NEDMANUALLY</td>
<td>no : false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARELABORSTANDARDSALL</td>
<td>&gt; &lt;</td>
<td>msdyn_arelaborstandardsall owed</td>
<td></td>
</tr>
<tr>
<td>OWED</td>
<td>no : false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>yes : true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSBUILDINGCOMPL</td>
<td>=</td>
<td>msdyn_primaryaddressbuild ingcompliment</td>
<td></td>
</tr>
<tr>
<td>MENT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSPOSTBOX</td>
<td>=</td>
<td>msdyn_primaryaddresspost box</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSSTREETNUMBER</td>
<td>=</td>
<td>msdyn_primaryaddressstreet number</td>
<td></td>
</tr>
<tr>
<td>AREWAREHOUSELOCATION</td>
<td>&gt; &lt;</td>
<td>msdyn_arewarehouselocation checkdigitsunique</td>
<td></td>
</tr>
<tr>
<td>CHECKDIGITSUNIQUE</td>
<td>yes : true</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>no : false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INVENTORYCOUNTINGREASONCODE</td>
<td>=</td>
<td>msdyn_inventorycountingreasoncodepolicyname</td>
<td></td>
</tr>
<tr>
<td>POLICYNAME</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUTOMATICUPDATESHIPPINGRULE</td>
<td>&gt; &lt;</td>
<td>msdyn_automaticupdateshippingrule</td>
<td></td>
</tr>
<tr>
<td></td>
<td>always : false</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>onQuantityDecrease : true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WAREHOUSERELEASERESERVATION</td>
<td>&gt; &lt;</td>
<td>msdyn_warehouserelasticsearchreservationrequirement</td>
<td></td>
</tr>
<tr>
<td>REQUIREMENTRULE</td>
<td>notapplicable : 192350000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>allowpartialreservation : 192350001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>requirefullreservation : 192350002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXTERNALLYLOCATEDWAREHOUSE</td>
<td>=</td>
<td>msdynExternallylocatedwarehousevendoraccountnumber</td>
<td></td>
</tr>
<tr>
<td>VENDORACCOUNTNUMBER</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INVENTORYSTATUSCHANGE</td>
<td>&gt; &lt;</td>
<td>msdyn_inventorystatuschange</td>
<td></td>
</tr>
<tr>
<td>RESERVATIONREMOVALLEVEL</td>
<td>notnone : 192350000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reservation : 192350001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>markingreservation : 192350002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Map Type</td>
<td>Customer Engagement Column</td>
<td>Default Value</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>----------</td>
<td>---------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>WAREHOUSEWORKPROCES SINGPOLICYNAME</td>
<td></td>
<td>msdyn_warehouseworkprocessingpolicyname</td>
<td></td>
</tr>
<tr>
<td>ISFALLBACKWAREHOUSE</td>
<td></td>
<td>msdyn_isfallbackwarehouse</td>
<td>Yes: true, No: false</td>
</tr>
<tr>
<td>ISFINANCIALNEGATIVERETAILSTOREINVENTORYALLOWED</td>
<td></td>
<td>msdyn_financialnegativestoreinventoryallowed</td>
<td>Yes: true, No: false</td>
</tr>
<tr>
<td>ISPALLETMOVEMENTDURINGCYCLECOUNTINGALLOWED</td>
<td></td>
<td>msdyn_palletmovementduringcyclecountingallowed</td>
<td>Yes: true, No: false</td>
</tr>
<tr>
<td>ISPHYSICALNEGATIVERETAILSTOREINVENTORYALLOWED</td>
<td></td>
<td>msdyn_physicalnegativestoreinventoryallowed</td>
<td>Yes: true, No: false</td>
</tr>
<tr>
<td>ISREFILLEDFROMMAINWAREHOUSE</td>
<td></td>
<td>msdyn_isrefilledfrommainwarehouse</td>
<td>Yes: true, No: false</td>
</tr>
<tr>
<td>ISRETAILSTOREWAREHOUSE</td>
<td></td>
<td>msdyn_isretailstorewarehouse</td>
<td>Yes: true, No: false</td>
</tr>
<tr>
<td>MAINREFILLINGWAREHOUSEID</td>
<td></td>
<td>msdyn_mainrefillingwarehouse.msdyn_warehouseidentifier</td>
<td></td>
</tr>
<tr>
<td>MASTERPLANNINGWORKCALENDARID</td>
<td></td>
<td>msdyn_masterplanningworkcalendarid</td>
<td></td>
</tr>
<tr>
<td>MAXIMUMBATCHPICKINGLISTQUANTITY</td>
<td></td>
<td>msdyn_maximumbatchpickinglistquantity</td>
<td></td>
</tr>
<tr>
<td>MAXIMUMPICKINGLISTLINEQUANTITY</td>
<td></td>
<td>msdyn_maximumpickinglistlinequantity</td>
<td></td>
</tr>
<tr>
<td>OPERATIONALSITEID</td>
<td></td>
<td>msdyn_operationalsite.msdyn_siteid</td>
<td></td>
</tr>
<tr>
<td>QUARANTINEWAREHOUSEID</td>
<td></td>
<td>msdyn_quarantinewarehouse.msdyn_warehouseidentifier</td>
<td></td>
</tr>
<tr>
<td>RETAILSTOREQUANTITYALLOCATIONSREPLENISMENTWEIGHT</td>
<td></td>
<td>msdyn_storeqtyallocationreplenishmentweight</td>
<td></td>
</tr>
<tr>
<td><strong>FINANCE AND OPERATIONS FIELD</strong></td>
<td><strong>MAP TYPE</strong></td>
<td><strong>CUSTOMER ENGAGEMENT COLUMN</strong></td>
<td><strong>DEFAULT VALUE</strong></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------</td>
<td>---------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>SHOULDWAREHOUSELOCATIONID</td>
<td>includeaisleid</td>
<td>msdyn_shouldwarehouselocationidincludeaisleid</td>
<td></td>
</tr>
<tr>
<td>TRANSITWAREHOUSEID</td>
<td>=</td>
<td>msdyn_transitwarehouseid</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEID</td>
<td>=</td>
<td>msdyn_warehousename</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSEID</td>
<td>&gt;</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSELOCATIONIDBINFORMAT</td>
<td>=</td>
<td>msdyn_warehouselocationidbinformat</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSELOCATIONIDRACKFORMAT</td>
<td>=</td>
<td>msdyn_warehouselocationidrackformat</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSELOCATIONIDSHELFFORMAT</td>
<td>=</td>
<td>msdyn_warehouselocationidshelfformat</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSENAME</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSESPECIFICDEFAULTINVENTORYSTATUSID</td>
<td>=</td>
<td>msdyn_warehousespecificdefaultinventorystatusid</td>
<td></td>
</tr>
<tr>
<td>WAREHOUSETYPE</td>
<td>&gt; &lt;</td>
<td>msdyn_warehousetype</td>
<td></td>
</tr>
<tr>
<td>ISPRIMARYADDRESSASSIGNED</td>
<td>&gt; &lt;</td>
<td>msdyn_isprimaryaddressassigned</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSCITY</td>
<td>=</td>
<td>msdyn_primaryaddresscity</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSCOUNTRYREGIONID</td>
<td>=</td>
<td>msdyn_primaryaddresscountrylettrregionid</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSCOUNTYID</td>
<td>=</td>
<td>msdyn_primaryaddresscountyid</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSDISTRICTNAME</td>
<td>=</td>
<td>msdyn_primaryaddressdistrictname</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSLATITUDE</td>
<td>=</td>
<td>msdyn_primaryaddresslatitude</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT FIELD</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------</td>
<td>---------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>PRIMARYADDRESSLONGITUDE</td>
<td>=</td>
<td>msdyn_primaryaddresslongi tude</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSLOCATIONROLES</td>
<td>=</td>
<td>msdyn_primaryaddresslocationroles</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSLOCATIONSALESTAXGROUPCODE</td>
<td>=</td>
<td>msdyn_primaryaddresslocationsalestaxgroupcode</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADRESSTYPEID</td>
<td>=</td>
<td>msdyn_primaryaddresstypeid</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSSTREET</td>
<td>=</td>
<td>msdyn_primaryaddressstreet</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSZIPCODE</td>
<td>=</td>
<td>msdyn_primaryaddresszipcode</td>
<td></td>
</tr>
<tr>
<td>EXTERNALLYLOCATEDWAREHOUSECUSTOMERACCOUNTNUMBER</td>
<td>=</td>
<td>msdynExternallylocatedwarehouselocation</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSCITYINKANA</td>
<td>=</td>
<td>msdyn_primaryaddresscityinkana</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSSTREETINKANA</td>
<td>=</td>
<td>msdyn_primaryaddressstreetinkana</td>
<td></td>
</tr>
<tr>
<td>PRIMARYADDRESSDESCRIPTION</td>
<td>=</td>
<td>msdyn_primaryaddressdescription</td>
<td></td>
</tr>
<tr>
<td>AREADVANCEDWAREHOUSEMANAGEMENTPROCESSENABLED</td>
<td>&lt; &gt;</td>
<td>msdyn_useadvancedwarehousemanagementprocesses</td>
<td></td>
</tr>
<tr>
<td>AREPICKINGLISTSDELIVERYMODESPECIFIC</td>
<td>&lt; &gt;</td>
<td>msdyn_arepickinglistsdeliverymodespecific</td>
<td></td>
</tr>
<tr>
<td>AREPICKINGLISTSSHIPMENTSPECIFICONLY</td>
<td>&lt; &gt;</td>
<td>msdyn_arepickinglistshipmentspecificonly</td>
<td></td>
</tr>
<tr>
<td>FORMATTEDPRIMARYADDRESS</td>
<td>=</td>
<td>msdyn_formattedprimaryaddress</td>
<td></td>
</tr>
<tr>
<td>ISBILLLOFLADINGPRINTINGBEFORESHIPEMPTIONENABLED</td>
<td>&lt; &gt;</td>
<td>msdyn_printbilllofladingbeforeshipment</td>
<td></td>
</tr>
<tr>
<td>RAWMATERIALPICKINGINVENTIONISSUESTATUS</td>
<td>&lt; &gt;</td>
<td>msdyn_rawmaterialpickinginventoryissuestatus</td>
<td></td>
</tr>
</tbody>
</table>
### Finance and Operations Field

<table>
<thead>
<tr>
<th>Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Will Automatic Load Reserve Inventory</td>
<td>&lt;</td>
<td>msdyn_willautomaticloadrel easeinventory</td>
<td></td>
</tr>
<tr>
<td>Will Inventory Status Change Remove Blocking</td>
<td>&lt;</td>
<td>msdyn_willinventorystatusch hangeremoveblocking</td>
<td></td>
</tr>
<tr>
<td>Will Manual Load Release Reserve Inventory</td>
<td>&lt;</td>
<td>msdyn_willmanualloadrelease inventory</td>
<td></td>
</tr>
<tr>
<td>Will Order Releasing Consolidate Shipments</td>
<td>&lt;</td>
<td>msdyn_willorderreleasingconsolidateshipments</td>
<td></td>
</tr>
<tr>
<td>Will Production Bom Reserve Warehouse Level Only</td>
<td>&lt;</td>
<td>msdyn_productionbomsreserve warehouselevel</td>
<td></td>
</tr>
<tr>
<td>Withholding Cancel Percentage Load Quantity</td>
<td>&lt;</td>
<td>msdyn_shippingcanceldecimalmentloadquantity</td>
<td></td>
</tr>
<tr>
<td>Will Warehouse Location ID Include Bin ID By Default</td>
<td>&lt;</td>
<td>msdyn_warehouselocationid include bin id</td>
<td></td>
</tr>
<tr>
<td>Will Warehouse Location ID Include Rack ID By Default</td>
<td>&lt;</td>
<td>msdyn_warehouselocationid include rack id by default</td>
<td></td>
</tr>
<tr>
<td>Will Warehouse Location ID Include Shelf ID By Default</td>
<td>&lt;</td>
<td>msdyn_warehouselocationid include shelf id</td>
<td></td>
</tr>
</tbody>
</table>

### Withholding Tax Codes (msdyn_withholdingtaxcodes)

This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Withholding Code</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>Withholding Tax Name</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
<tr>
<td>Withholding Tax Round Off</td>
<td>=</td>
<td>msdyn_roundoff</td>
<td></td>
</tr>
</tbody>
</table>
Withholding tax groups (msdyn_withholdingtaxgroups)
This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WithholdingTaxGroupCode</td>
<td>=</td>
<td>msdyn_name</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>=</td>
<td>msdyn_description</td>
<td></td>
</tr>
</tbody>
</table>

Worker (cdm_workers)
This template synchronizes data between Finance and Operations apps and Dataverse.

<table>
<thead>
<tr>
<th>Finance and Operations Field</th>
<th>Map Type</th>
<th>Customer Engagement Column</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PersonnelNumber</td>
<td>=</td>
<td>cdm_workernumber</td>
<td></td>
</tr>
<tr>
<td>Firstname</td>
<td>=</td>
<td>cdm_firstname</td>
<td></td>
</tr>
<tr>
<td>Middlename</td>
<td>=</td>
<td>cdm_middlename</td>
<td></td>
</tr>
<tr>
<td>Lastname</td>
<td>=</td>
<td>cdm_lastname</td>
<td></td>
</tr>
<tr>
<td>Workertype</td>
<td>&gt;&gt;</td>
<td>cdm_type</td>
<td></td>
</tr>
<tr>
<td>Employed</td>
<td>=</td>
<td>cdm_type</td>
<td></td>
</tr>
<tr>
<td>Pending</td>
<td>=</td>
<td>cdm_status</td>
<td></td>
</tr>
<tr>
<td>Terminated</td>
<td>=</td>
<td>cdm_status</td>
<td></td>
</tr>
<tr>
<td>PrimaryContactEmail</td>
<td>=</td>
<td>cdm_primaryemailaddress</td>
<td></td>
</tr>
<tr>
<td>PrimaryContactPhone</td>
<td>=</td>
<td>cdm_primarytelephone</td>
<td></td>
</tr>
<tr>
<td>FINANCE AND OPERATIONS FIELD</td>
<td>MAP TYPE</td>
<td>CUSTOMER ENGAGEMENT COLUMN</td>
<td>DEFAULT VALUE</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>-----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>PRIMARYCONTACTFACEBOOK</td>
<td>=</td>
<td>cdm_facebookidentity</td>
<td></td>
</tr>
<tr>
<td>PRIMARYCONTACTTWITTER</td>
<td>=</td>
<td>cdm_twitteridentity</td>
<td></td>
</tr>
<tr>
<td>PRIMARYCONTACTLINKEDIN</td>
<td>=</td>
<td>cdm_linkedinidentity</td>
<td></td>
</tr>
<tr>
<td>PRIMARYCONTACTURL</td>
<td>=</td>
<td>cdm_websiteurl</td>
<td></td>
</tr>
<tr>
<td>GENDER</td>
<td>&gt; &lt;</td>
<td>cdm_gender</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>754400000 male</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>754400001 female</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>754400003 nonSpecific</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>754400002 none</td>
<td></td>
</tr>
<tr>
<td>BIRTHDATE</td>
<td>=</td>
<td>cdm_birthdate</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>&gt;</td>
<td>cdm fullname</td>
<td></td>
</tr>
</tbody>
</table>
Support for Field Service solutions

Microsoft supports dual-write on top of existing Dataverse environments that are based on Field Service solutions.

For more information, see Integrate Dynamics 365 Field Service and Supply Chain Management.

Support for Project Service Automation solutions

Microsoft supports dual-write on top of existing Dataverse environments that are based on Project Service Automation solutions.
To migrate your Prospect to cash data from Data Integrator to dual-write, follow these steps.

1. Run the Prospect to cash Data Integrator jobs to do one final full synchronization. In this way, you ensure that both systems (Finance and Operations apps and customer engagement apps) have all the data.

2. To help prevent potential data loss, export the Prospect to cash data from Microsoft Dynamics 365 Sales to an Excel file or a comma-separated values (CSV) file. Export data from the following entities:
   - Account
   - Contact
   - Invoice
   - Invoice products
   - Order
   - Order products
   - Products
   - Quote
   - Quote products

3. Uninstall the Prospect to cash solution from the Sales environment. This step removes the columns and corresponding data that the Prospect to cash solution introduced.

4. Install the dual-write solution.

5. Create a dual-write connection between the Finance and Operations app and the customer engagement app for one or more legal entities.

6. Enable dual-write table maps, and run the initial synchronization for the required reference data. (For more information, see Considerations for initial synchronization.) Examples of required data include customer groups, payment terms, and payment schedules. Don't enable the dual-write maps for tables that require initialization, such as the account, quote, quote line, order, and order line tables.

7. In the customer engagement app, go to Advanced Settings > System Settings > Data Management > Duplicate detection rules, and disable all the rules.

8. Initialize the tables that are listed in step 2. For instructions, see the remaining sections of this topic.

9. Open the Finance and Operations app, and enable the table maps, such as the account, quote, quote line, order, and order line table maps. Then run the initial synchronization. (For more information, see Considerations for initial synchronization.) This process will sync additional information from the Finance and Operations app, such as processing status, shipping and billing addresses, sites, and warehouses.

**Account table**

1. In the **Company** column, enter the company name, such as **USMF**.

2. In the **Relationship Type** column, enter **Customer** as a static value. You might not want to classify every account record as a customer in your business logic.

3. In the **Customer Group ID** column, enter the customer group number from the Finance and Operations
app. The default value from the Prospect to cash solution is 10.

4. If you're using the Prospect to cash solution without any customization of Account Number, enter an Account Number value in the Party Number column. If there are customizations, and you don't know the party number, pull this information from the Finance and Operations app.

Contact table

1. In the Company column, enter the company name, such as USMF.

2. Set the following columns, based on the IsActiveCustomer value in the CSV file:
   - If IsActiveCustomer is set to Yes in the CSV file, set the Sellable column to Yes. In the Customer Group ID column, enter the customer group number from the Finance and Operations app. The default value from the Prospect to cash solution is 10.
   - If IsActiveCustomer is set to No in the CSV file, set the Sellable column to No, and set the Contact For column to Customer.

3. If you're using the Prospect to cash solution without any customization of Contact Number, set the following columns:
   - Migrate the contact number from the CSV file (msdynce_contactnumber) to the contact number in the Contact table (msdyn_contactnumber).
   - Use values from the Contact Number table in the Party Number column.
   - Use values from the Contact Number table in the Account Number/Contact Person ID column.

Invoice table

Because data from the Invoice table is designed to flow one way, from the Finance and Operations app to the customer engagement app, initialization isn't required. Run the initial synchronization to migrate all the required data from the Finance and Operations app to the customer engagement app. For more information, see Considerations for initial synchronization.

Order table

1. In the Company column, enter the company name, such as USMF.

2. Copy the value of the Order ID column in the CSV file to the Sales Order Number column.

3. Copy the value of the Customer column in the CSV file to the Invoice customer number column.

4. Copy the value of the Ship To Country/Region column in the CSV file to the Ship To Country/Region column. Examples of this value include US and United States.

5. Set the Requested Receipt Date column. If you aren't using a receipt date, use the Requested Delivery Date, Date Fulfilled, and Date Submitted columns in the CSV file. An example of this value is 2020-03-27T00:00:00Z.

6. Set the Language column. An example of this value is en-us.

7. Set the Order Type column by using the Item-based column.

Order products table

- In the Company column, enter the company name, such as USMF.

Products table

Because data from the Products table is designed to flow one way, from the Finance and Operations app to the customer engagement app, initialization isn't required. Run the initial synchronization to migrate all the required data from the Finance and Operations app to the customer engagement app. For more information, see
Considerations for initial synchronization.

Quote and Quote product tables

For the Quote table, follow the instructions in the Order table section earlier in this topic. For the Quote product table, follow the instructions in the Order products table section.
This topic provides general troubleshooting information for dual-write integration between Finance and Operations apps and Dataverse.

**IMPORTANT**
Some of the issues that this topic addresses might require either the system admin role or Microsoft Azure Active Directory (Azure AD) tenant admin credentials. The section for each issue explains whether a specific role or credentials are required.

---

Enable and view the plug-in trace log in Dataverse to view error details

**Required role to turn on the trace log and view errors:** System admin

To turn on the trace log, follow these steps.

1. Sign in to the customer engagement app, open the **Settings** page, and then, under **System**, select **Administration**.
2. On the **Administration** page, select **System Settings**.
3. On the **Customization** tab, in the **Plug-in and custom workflow activity tracing** column, select **All** to enable the plug-in trace log. If you want to log trace logs only when exceptions occur, you can select **Exception** instead.

To view the trace log, follow these steps.

1. Sign in to the customer engagement app, open the **Settings** page, and then, under **Customization**, select **Plug-in Trace Log**.
2. Find the trace logs where the **Type Name** column is set to **Microsoft.Dynamics.Integrator.DualWriteRuntime.Plugins.PreCommmitPlugin**.
3. Double-click an item to view the full log, and then, on the **Execution** FastTab, review the **Message Block** text.

---

Enable debug mode to troubleshoot live synchronization issues in Finance and Operations apps

**Required role to view the errors:** System admin

Dual-write errors that originate in Dataverse can appear in the Finance and Operations app. To enable verbose logging for the errors, following these steps:

1. For all project configurations in Finance and Operations app there is a flag **IsDebugMode** on the **DualWriteProjectConfiguration** table.
2. Open the **DualWriteProjectConfiguration** using the Excel addin. To use the addin, enable design mode in the Finance and Operations Excel addin and add the **DualWriteProjectConfiguration** to the sheet. For more information, see View and update entity data with Excel.
3. Set **IsDebugMode** to Yes on the project.
4. Run the scenario that is generating errors.
5. The verbose logs are stored in the **DualWriteErrorLog** table.

6. To lookup data on table browser use the following link:
   
   https://999aos.cloudax.dynamics.com/?mi=SysTableBrowser&tableName=DualWriteErrorLog, replacing **999** as needed.

7. Update again after **KB 4595434**, which is available for platform updates 37 and later. If you have this fix installed then the debug mode will capture more logs.

**Check synchronization errors on the virtual machine for the Finance and Operations app**

**Required role to view the errors:** System administrator

1. Sign in to Microsoft Dynamics Lifecycle Services (LCS).
2. Open the LCS project that you chose to do the dual-write testing for.
3. Select the **Cloud-hosted environments** tile.
4. Use Remote Desktop to sign in to the virtual machine (VM) for the Finance and Operations app. Use the local account that is shown in LCS.
5. Open Event viewer.
6. Select **Applications and Services Logs > Microsoft > Dynamics > AX-DualWriteSync > Operational**.
7. Review the list of recent errors.

**Unlink and link another Dataverse environment from a Finance and Operations app**

**Required role to unlink the environment:** System administrator for either Finance and Operations app or Dataverse.

1. Sign in to the Finance and Operations app.
2. Go to **Workspaces > Data management**, and select the **Dual Write** tile.
3. Select all running mappings, and then select **Stop**.
4. Select **Unlink environment**.
5. Select **Yes** to confirm the operation.

You can now link a new environment.

**Unable to view the sales order line Information form**

When you create a sales order in Dynamics 365 Sales, clicking on **+ Add products** might redirect you to the Dynamics 365 Project Operations order line form. There is no way from that form to view the sales order line **Information** form. The option for **Information** does not appear in the dropdown below **New Order Line**. This happens because Project Operations has been installed in your environment.

To re-enable the **Information** form option, follow these steps:

1. Navigate to the **Order Line** table.
2. Find the **Information** form under the forms node.
3. Select the **Information** form and click **Enable security roles**.
4. Change the security setting to **Display to everyone**.

**How to enable and save network trace so that traces can be attached**
to support tickets

The support team might need to review network traces to troubleshoot some issues. To create a network track, follow these steps:

**Chrome**

1. In the opened tab, press F12 or choose Developer tools to open the developer tools.
2. Open the Network tab and type integ in the filter text box.
3. Run your scenario and observe the requests being logged.
4. Right-click on the entries and select Save all as a HAR with content.

**Microsoft Edge**

1. In the opened tab, press F12 or choose Developer tools to open the developer tools.
2. Open the Network tab.
3. Run your scenario.
4. Select save to export the results as HAR.
This topic provides troubleshooting information for dual-write integration between Finance and Operations apps and Dataverse. Specifically, it provides information that can help you fix issues that might occur during the initial setup of dual-write integration.

**IMPORTANT**

Some of the issues that this topic addresses might require either the system admin role or Microsoft Azure Active Directory (Azure AD) tenant admin credentials. The section for each issue explains whether a specific role or credentials are required.

---

**You can't link a Finance and Operations app to Dataverse**

**Required role to set up dual-write:** System administrator in Finance and Operations apps and Dataverse.

Errors on the Setup link to Dataverse page are usually caused by incomplete setup or permissions issues. Make sure that the whole health check passes on the Setup link to Dataverse page, as shown in the following illustration. You can't link dual-write unless the whole health check passes.

---

You must have Azure AD tenant admin credentials to link the Finance and Operations and Dataverse environments. After you link the environments, users can sign in by using their account credentials and update an existing table map.
Find the limit on the number of legal tables or companies that can be linked for dual-write

You might receive the following error message when you try to enable maps:

_Dual write failure - Plugin registration failed: [Unable to get partition map for project DWM-1ae35e60-4bc2-4905-88ea-69efd3b29260-7f12cb89-1550-42e2-858e-4761fc1443ea. Error Exceeds the maximum partitions allowed for mapping DWM-1ae35e60-4bc2-4905-88ea-69efd3b29260-7f12cb89-1550-42e2-858e-4761fc1443ea]_. One or more errors occurred.

The current limit when you link the environments is approximately 40 legal tables. This error occurs if you try to enable maps, and more than 40 legal tables are linked between the environments.

**Connection set failed while linking environment**

While linking the dual-write environment, the action fails with an error message:

_Saving connection set failed! An item with the same key has already been added._

_Dual-write does not support multiple legal entities/companies with the same name. For example, If you have two companies with "DAT" name in the Dataverse then it will get this error message._

To unblock the customer, remove duplicate records from _cdm_company_ table in Dataverse. Also, if the _cdm_company_ table has records with blank name, remove or correct those records.

**Error when opening the Dual-write page in Finance and Operations apps**

You might receive the following error message when you try to link a Dataverse environment for dual-write:

_Response status code does not indicate success: 404 (Not Found)._

This error occurs when the app consent step is not complete. You can validate if consent has been provided by logging on to _portal.azure.com_ using the tenant admin account, and check if the 3rd party app with ID _33976c19-1db5-4c02-810e-c243db79efde_ shows up in AAD’s Enterprise applications list. If not, then rerun the consent step as described in the next section.

**Providing App consent**

- Launch the following URL with your admin credentials.

  https://login.microsoftonline.com/common/oauth2/authorize?client_id=33976c19-1db5-4c02-810e-c243db79efde&response_type=code&prompt=admin_consent

- Select _Accept_ to consent. You are providing the consent to install the app (with _id=33976c19-1db5-4c02-810e-c243db79efde_) in your tenant.

- This app is required for Dataverse to communicate to Finance and Operations apps.
Finance and Operations environment is not discoverable

You might receive the following error message:

*Finance and Operations apps environment ***.cloudax.dynamics.com is not discoverable.*

There are two things that can cause an issue with environment not being discoverable:

- The user used for login is not in the same tenant as the Finance and Operations instance.
- There are some legacy Finance and Operations instances that were Microsoft-hosted that had an issue with discovery. To fix this, update the Finance and Operations instance. The environment becomes discoverable with any update.
This topic provides troubleshooting information for dual-write integration between Finance and Operations apps and Dataverse. Specifically, it provides information that can help you fix issues that might occur during initial synchronization.

**IMPORTANT**

Some of the issues that this topic addresses might require either the system admin role or Microsoft Azure Active Directory (Azure AD) tenant admin credentials. The section for each issue explains whether a specific role or credentials are required.

Check for initial synchronization errors in a Finance and Operations app

After you enable the mapping templates, the status of the maps should be Running. If the status is Not running, errors occurred during initial synchronization. To view the errors, select the Initial sync details tab on the Dual-write page.

You can't complete initial synchronization: 400 Bad Request

**Required role to fix the issue:** System admin

You might receive the following error message when you try to run the mapping and initial synchronization:

```
[(Bad Request), The remote server returned an error: (400) Bad Request), AX export encountered an error.
```

Here is an example of the full error message.
Dual write Initial Sync completed with status: Error. Following are the details:

Executed leg: From AX Financial dimensions to CRM msdyn_dimensionattributes
with exported records count: 0, ImportRecordsErrorCount: 0, ImportRecordsInsertedCount: 0 and ImportRecordsUpdatedCount: 0

ErrorsDetails:
Dual write Initial sync failed
Message: ([Bad Request], The remote server returned an error: (400) Bad Request.), AX export encountered an error

Stacktrace: at
in X:\bt\1024532\repo\src\Core\QueryGenerator\AxClient.cs:line 265
--- End of stack trace from previous location where exception was thrown ---
at System.Runtime.ExceptionServices.ExceptionDispatchInfo.Throw()
at System.Runtime.CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)
The error messages that you receive will have the following form.

*Couldn't resolve the guid for the field: <field>. The lookup was not found: <value>.* Try this URL(s) to check if the reference data exists:

```
https://focdsdevtest2.crm.dynamics.com/api/data/v9.0/entity?$select=<field>&$filter=<field> eq <value>
```

Here are some examples:

- *Couldn't resolve the guid for the field: msdyn_vendorprimarycontactperson.msdyn_contactpersonid. The lookup was not found: 000056.* Try this URL(s) to check if the reference data exists:
  
  ```
  https://focdsdevtest2.crm.dynamics.com/api/data/v9.0/contacts?
  $select=msdyn_contactpersonid/contactid&$filter=msdyn_contactpersonid eq '000056'
  ```

- *Couldn't resolve the guid for the field: msdyn_invoicevendoraccountnumber.msdyn_vendoraccountnumber. The lookup was not found: V24-1.* Try this URL(s) to check if the reference data exists:
  
  ```
  https://focdsdevtest2.crm.dynamics.com/api/data/v9.0/msdn_vendors?
  $select=msdyn_vendoraccountnumber,msdyn_vendorid&$filter=msdyn_vendoraccountnumber eq 'V24-1'
  ```

If any rows in the vendor table have values in the `PrimaryContactPersonId` and `InvoiceVendorAccountNumber` columns, follow these steps to complete the initial synchronization.

1. In the Finance and Operations app, delete the `PrimaryContactPersonId` and `InvoiceVendorAccountNumber` columns from the mapping, and then save the mapping.

   a. On the dual-write mapping page for **Vendors V2 (msdyn_vendors)**, on the **Table mappings** tab, in the left filter, select **Finance and Operations apps.Vendors V2**. In the right filter, select **Sales.Vendor**.

   b. Search for `primarycontactperson` to find the `PrimaryContactPersonId` source column.

   c. Select **Actions**, and then select **Delete**.

   d. Repeat these steps to delete the `InvoiceVendorAccountNumber` column.

   e. Save your changes to the mapping.

2. Turn off change tracking for the **Vendors V2** table.

   a. In the **Data management** workspace, select the **Data tables** tile.

   b. Select the **Vendors V2** table.

   c. On the Action Pane, select **Options**, and then select **Change tracking**.
d. Select **Disable Change Tracking**.

3. Run initial synchronization for the **Vendors V2 (msdyn_vendors)** mapping. The initial synchronization should run successfully, without any errors.

4. Run initial synchronization for the **CDS Contacts V2 (contacts)** mapping. You must sync this mapping if you want to sync the primary contact column on the vendors table, because initial synchronization must also be done for the contact rows.

5. Add the **PrimaryContactPersonId** and **InvoiceVendorAccountNumber** columns back to the **Vendors V2 (msdyn_vendors)** mapping, and then save the mapping.

6. Run initial synchronization again for the **Vendors V2 (msdyn_vendors)** mapping. Because change tracking is turned off, all the rows will be synced.

7. Turn change tracking back on for the **Vendors V2** table.

**Resolve errors in the Customers V3–to–Accounts table mapping**

You might encounter initial synchronization errors for the mapping of **Customers V3** to **Accounts** if the tables have existing rows where there are values in the **ContactPersonID** and **InvoiceAccount** columns. These errors occur because **InvoiceAccount** is a self-referencing column, and **ContactPersonID** is a circular reference in the vendor mapping.

The error messages that you receive will have the following form.

*Couldn't resolve the guid for the field: <field>. The lookup was not found: <value>. Try this URL(s) to check if the reference data exists:*

```
https://focdsdevtest2.crm.dynamics.com/api/data/v9.0/<entity>?$select=<field>&$filter=<field> eq <value>
```

Here are some examples:

- *Couldn’t resolve the guid for the field: primarycontactid.msdyn_contactpersonid. The lookup was not found: 000056. Try this URL(s) to check if the reference data exists:*

  ```
  https://focdsdevtest2.crm.dynamics.com/api/data/v9.0/contacts?$select=msdyn_contactpersonid.contactid&$filter=msdyn_contactpersonid eq '000056'
  ```

- *Couldn’t resolve the guid for the field: msdyn_billingaccount.accountnumber. The lookup was not found: 1206-1. Try this URL(s) to check if the reference data exists:*

  ```
  https://focdsdevtest2.crm.dynamics.com/api/data/v9.0/accounts?$select=accountnumber.account&$filter=accountnumber eq '1206-1'
  ```
If any rows in the customer table have values in the ContactPersonID and InvoiceAccount columns, follow these steps to complete the initial synchronization. You can use this approach for any out-of-box tables, such as Accounts and Contacts.

1. In the Finance and Operations app, delete the ContactPersonID and InvoiceAccount columns from the Customers V3 (accounts) mapping, and then save the mapping.
   b. Search for contactperson to find the ContactPersonID source column.
   c. Select Actions, and then select Delete.
   d. Repeat these steps to delete the InvoiceAccount column.
   e. Save your changes to the mapping.

2. Turn off change tracking for the Customers V3 table.
   a. In the Data management workspace, select the Data tables tile.
   b. Select the Customers V3 table.
   c. On the Action Pane, select Options, and then select Change tracking.
   d. Select Disable Change Tracking.
3. Run initial synchronization for the Customers V3 (Accounts) mapping. The initial synchronization should run successfully, without any errors.

4. Run initial synchronization for the CDS Contacts V2 (contacts) mapping.

   **NOTE**
   There are two maps that have the same name. Be sure to select the map that has the following description on the Details tab: Dual-write template for sync between FO.CDS Vendor Contacts V2 to CDS.Contacts. Requires new package [Dynamics365SupplyChainExtended].

5. Add the InvoiceAccount and ContactPersonId columns back to the Customers V3 (Accounts) mapping, and then save the mapping. Both the InvoiceAccount column and the ContactPersonId column are now part of live synchronization mode again. In the next step, you will do the initial synchronization for these columns.

6. Run initial synchronization again for the Customers V3 (Accounts) mapping. Because change tracking is turned off, the data for InvoiceAccount and ContactPersonId will be synced from the Finance and Operations app to Dataverse.

7. To sync the data for InvoiceAccount and ContactPersonId from Dataverse to the Finance and Operations app, you must use a data integration project.

   a. In Power Apps, create a data integration project between the Sales.Account and Finance and Operations apps.Customers V3 tables. The data direction must be from Dataverse to the Finance and Operations app. Because InvoiceAccount is a new attribute in dual-write, you might want to skip initial synchronization for it. For more information, see Integrate data into Dataverse.

   The following illustration shows a project that updates CustomerAccount and ContactPersonId.

   ![Project Illustration]

   b. Add the company criteria in the filter on the Dataverse side, so that only rows that match the filter criteria will be updated in the Finance and Operations app. To add a filter, select the filter button. Then, in the Edit query dialog box, you can add a filter query such as _msdyn_company_value eq '<guid>'.
If you don't enter a filter query for `_msdyn_company_value`, all the rows will be synced.

The initial synchronization of the rows is now completed.

8. In the Finance and Operations app, turn change tracking back on for the Customers V3 table.

**Initial sync failures on maps with more than 10 lookup fields**

You might receive the following error message when you try run an initial sync failures on Customers V3 - Accounts, Sales orders mappings, or any map with more than 10 lookup fields:

`CRMExport: Package execution complete. Error Description 5 Attempts to get data from https://xxxxx//datasets/yyyyy/tables/accounts/items?$select=accountnumber, address2_city, address2_country,... (msdyn_company/cdm_companyid eq 'id')&$orderby=accountnumber asc failed.`

Because of the lookup limitation on the query, the initial sync fails when the entity mapping contains more than 10 lookups. For more information, see [Retrieve related table records with a query](#).

To fix this issue, follow these steps:

1. Remove optional lookup fields from the dual-write entity map so that the number of lookups is 10 or fewer.
2. Save the map and do the initial sync.
3. When the initial sync for the first step is successful, add the remaining lookup fields and remove the lookup fields that you synced in first step. Make sure that the number of lookup fields is 10 or fewer. Save the map and run the initial sync.
4. Repeat these steps until all the lookup fields are synced.
5. Add all the lookup fields back to the map, save the map, and run the map with Skip initial sync.

This process enables the map for live sync mode.

**Known issue during initial sync of Party postal addresses and party electronic addresses**

You might receive the following error message when you try to run the initial syn of Party postal addresses and party electronic addresses:

`Party number could not found in Dataverse.`
There is a range set on `DirPartyCDSEntity` in Finance and Operations apps that filters parties of type `Person` and `Organization`. As a result, an initial sync of the `CDS Parties – msdyn_parties` mapping will not sync parties of other types, including `Legal Entity` and `Operating Unit`. When the initial sync runs for `CDS Party postal addresses (msdyn_partypostaladdresses)` or `Party Contacts V3 (msdyn_partyelectronicaddresses)` you might receive the error.

We are working on a fix to remove the party type range on the Finance and Operations entity so that parties of all types can synchronize to Dataverse successfully.

**Are there any performance issues while running initial sync for Customers or Contacts data?**

If you have run the initial sync for `Customer` data and have the `Customer` maps running and then you are run the initial sync for `Contacts` data, there might be performance issues during inserts and updates to the `LogisticsPostalAddress` and `LogisticsElectronicAddress` tables for `Contact` addresses. The same global postal address and electronic address tables are tracked for `CustCustomerV3Entity` and `VendVendorV2Entity` and dual-write tries to build more queries to write data to other side. If you have already run the initial sync for `Customer`, then stop the corresponding map while running initial sync for `Contacts` data. Do the same thing for the `Vendor` data. When the initial sync is finished, you can run all the maps by skipping the initial sync.
This topic provides troubleshooting information for dual-write integration between Finance and Operations apps and Microsoft Dataverse. Specifically, it provides information that can help you fix issues with live synchronization.

**IMPORTANT**

Some of the issues that this topic addresses might require either the system admin role or Azure Active Directory (Azure AD) tenant admin credentials. Each section explains whether a specific role or specific credentials are required.

---

**Live synchronization shows an error when you create a row**

You might receive the following error message when you create a row in a Finance and Operations app:

```javascript

{"error":{"code":"0x80072560","message":"The user is not a member of the organization."}, The remote server returned an error: (403) Forbidden.

```

To fix the issue, follow the steps in System requirements and prerequisites. To complete those steps, dual-write application users who were created in Dataverse must have the system admin role. The default owning team must also have the system admin role.

**Required role to fix the issue:** System admin

---

**Live synchronization shows an error when you try to save table data**

You might receive the following error message when you try to save table data in a Finance and Operations app:

```

Cannot save the changes to the database. Unit of Work can not commit transaction. Unable to write data to entity uoms. Writes to UnitOfMeasureEntity failed with error message Unable to sync with entity uoms.

```

To fix the issue, make sure that prerequisite reference data exists in both the Finance and Operations app and Dataverse. For example, if a customer record belongs to a specific customer group, make sure that the customer group record exists in Dataverse.

If data exists in both places, and you’ve confirmed that the issue isn’t data related, follow these steps.

1. Open the DualWriteProjectConfigurationEntity entity by using the Excel add-in. To use the add-in, enable design mode in the Finance and Operations Excel add-in, and add DualWriteProjectConfigurationEntity to a worksheet. For more information, see View and update entity data with Excel.

2. Select and delete the records that have issues in the dual-write map and project. There will be two records for every dual-write mapping.

3. Publish the changes by using the Excel add-in. This step is important because it deletes the records from the entity and underlying tables.

---

**Handle read or write privilege errors when you create data in a Finance and Operations app**

You might receive a “Bad Request” error message when you create data in a Finance and Operations app.
To fix the issue, you must enable the missing privilege by assigning the correct security role to the team of mapped Dynamics 365 Sales or Dynamics 365 Customer Service business units.

1. In the Finance and Operations app, find the business unit that is mapped in the Data Integration connection set.

2. In the customer engagement app, sign in to the environment, go to Setting > Security, and find the team of the mapped business unit.

3. Open the page for the team for editing, and then select Manage roles.

4. In the Manage Team Roles dialog box, assign the role that has the read/write privilege for the relevant
Fix synchronization issues in an environment that has a recently changed Dataverse environment

**Required role to fix the issue:** System admin

You might receive the following error message when you create data in a Finance and Operations app:

```json
{ "entityName": "CustCustomerV3Entity", "executionStatus": 2, "fieldResponses": [], "recordResponses": [{ "errorMessage": "Unable to generate payload for entity CustCustomerV3Entity", "logDateTime": "2019-08-27T18:51:52.5843124Z", "verboseError": "Payload creation failed with error Invalid URI: The URI is empty."
}, { "isErrorCountUpdated": true }]}
```

Here is the error message in the customer engagement app:

```
An unexpected error occurred from ISV code. (ErrorType = ClientError) Unexpected exception from plug-in (Execute): Microsoft.Dynamics.Integrator.DualWriteRuntime.Plugins.PostCommitPlugin: System.Exception: failed to process entity account - (A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond.
```

This error occurs if the Dataverse environment is incorrectly reset when you try to create data in the Finance and Operations app.

**IMPORTANT**

If you’ve relinked the environments, you must stop all the entity maps before you continue with the mitigation steps.

To fix the issue, you must complete steps in both Dataverse and the Finance and Operations app.

1. In the Finance and Operations app, follow these steps:
   a. Open the DualWriteProjectConfigurationEntity entity by using the Excel add-in. To use the add-in, enable design mode in the Finance and Operations Excel add-in, and add DualWriteProjectConfigurationEntity to a worksheet. For more information, see View and update entity data with Excel.
   b. Select and delete the records that have issues in the dual-write map and project. There will be two records for every dual-write mapping.
   c. Publish the changes by using the Excel add-in. This step is important because it deletes the records from the entity and underlying tables.
   d. To help prevent errors when you relink the Finance and Operations or Dataverse environments, make sure that no dual-write configurations remain.

2. In Dataverse, follow these steps:
   a. Sign in to your Dataverse environment (for example, [https://****.crm.dynamics.com/](https://****.crm.dynamics.com/) ).
   b. Go to Advanced Settings > Advanced Find.
   c. Select DualWrite Runtime Configuration.
   d. Select the column to view.
   e. Select Results to view the configurations.
   f. Delete all the instances.

3. In the Finance and Operations app, follow these steps:
Live synchronization error after you do a full database copy

You might receive the following error message after you run a full database copy from one system to another and then try to run a database operation:

SecureConfig Organization (???) does not match actual CRM Organization (???).

The error message is shown from the dual-write runtime plug-in to ensure that the dual-write configuration that is set up in one system can't be used in another system.

To fix the issue, delete all the records in the msdyn_dualwriteruntimeconfig table after you restore the database. For more information, see Unlink and relink dual-write environments.

Live synchronization issues that are caused by incorrect query filter syntax on the dual-write maps

Even though the query expression for a dual-write map filter is syntactically correct, it might not work as expected. The filter expression is on an entity, not on an individual data source of a query object. Therefore, the SQL query that is generated doesn't return the expected results.

Here is an example.

You might expect projects that have no parent to be filtered out. However, the filter doesn't work because it's translated to a query that resembles the following example.

```
SELECT T1.RECID,T1.MODIFIEDDATETIME,T1.RECVERSION,T1.RECID,T1.DIMENSION,
   T1.LOCATION,T1.PROJECTCONTROLLER,T1.PROJECTID,T1.PROJECTMANAGER,T1.REFERENCE,
   T1.SALESMANAGER,T1.SCHEDULED,T1.RECVERSION#8,T1.RECVERSION#7,
   T1.RECVERSION#6,T1.RECVERSION#5,T1.RECVERSION#4,T1.RECVERSION#3,
   T1.RECVERSION#2,T1.RECID#8,T1.RECID#7,T1.RECID#6,T1.RECID#5,
   T1.RECID#4,T1.RECID#3,T1.RECID#2,T1.PARTITION FROM PROJECTENTITY T1
WHERE((((((((((((PARTITION=5637144576) AND (DATAAREAID=N'usmf')) AND
   (PARTITION#2=5637144576) OR (PARTITION#2 IS NULL)) AND
   (PARTITION#3=5637144576) OR (PARTITION#3 IS NULL)) AND
   (PARTITION#4=5637144576) OR (PARTITION#4 IS NULL)) AND
   (PARTITION#5=5637144576) OR (PARTITION#5 IS NULL)) AND
   (PARTITION#6=5637144576) OR (PARTITION#6 IS NULL)) AND
   (PARTITION#7=5637144576) OR (PARTITION#7 IS NULL)) AND
   (PARTITION#8=5637144576) OR (PARTITION#8 IS NULL)) AND
   (PARENTPROJECT=''))
ORDER BY T1.PROJECTID
```
The actual result is that the `ParentProject` field is evaluated to `null`. However, `null` isn't the same as the empty string. Because of this mismatch, the query filter doesn't return valid results.

To fix the issue, follow these steps.

1. Add a computed column that can be added in an extension model, and that is backed by logic that converts `null` to the empty string.

   ```x++
   SysComputedColumn::if(SysComputedColumn::isNullExpression(ParentProject),
   SysComputedColumn::returnLiteral(""), fieldName);
   ```

2. Use the filter on the new computed column instead of the default column.

To evaluate the filter in a development environment, you can use following X++ code to validate the results. Run this code as a standalone program. You can use it to evaluate different kinds of filters that are applicable for an entity before you use those filters on dual-write maps. The query can be run against the database to evaluate discrepancies.

```x++
var entityName = "PROJECTENTITY";
var filterExpression = '(ParentProject == ")
Query query = new Query();
query.literals(NoYes::Yes);
QueryBuildDataSource qbd = query.addDataSource tablename2Id(entityName));
qbd.addRange(fieldname2id(qbd.table(), identifierStr(RecVersion))).value(filterExpression);
qbd.addSelectionField(fieldname2id(qbd.table(), identifierStr(RecId)));
QueryRun qRun = new QueryRun(query);
// This provides the actual sql statement to execute
var actualSqlStatement = query.getSQLStatement();
while(qRun.next())
{
    var rec = qRun.get(tableName2Id(entityName));
}
```

Data from Finance and Operations apps isn't synced to Dataverse

During live synchronization, you might encounter an issue where only part of the data is synced from Finance and Operations apps to Dataverse, or data isn't synced at all.

**NOTE**
You must fix this issue during development.

Before you start to fix the issue, review the following prerequisites:

- Make sure that your custom changes are written in a single transaction scope.
- Business events and the dual-write framework don't handle `doinsert()`, `doUpdate()`, and `recordset()` operations, or records where `skipBusinessEvents(true)` is marked. If your code is inside these functions, dual-write won't be triggered.
- Business events must be registered for the data source that is mapped. Some data sources might use an outer join and might be marked as read only in Finance and Operations apps. These data sources aren't tracked.
- Changes will be triggered only if the modifications are on the mapped fields. Unmapped field modifications won't trigger dual-write.
- Make sure that filter evaluations provide a valid result.

**Troubleshooting steps**
1. Review field mappings on the dual-write admin page. If a field isn't mapped from Finance and Operations apps to Dataverse, it won't be tracked. For example, in the following illustration, the **Description** field is tracked from Dataverse, but not from Finance and Operations apps. No changes to that field inside Finance and Operations apps will be tracked.

![Image of field mappings](image1.png)

2. Determine whether the data source is tracked in the business events definition. For example, in the following illustration, no field from the **DefaultDimensionDAVs** table and underlying tables will be tracked for changes. Data sources that use an outer join and that are marked as read only aren't tracked.

![Image of business events definition](image2.png)

3. Determine whether the mapped table fields appear in the **BUSINESSEVENTSDEFINITION** table, as shown in the following illustration. If you don't find the field that you're looking for in the query result, it won't be triggered by dual-write.

![Image of query result](image3.png)
Sample scenario

In Finance and Operations apps, there is an update to the address for a contact record, but the address change isn’t synced to Dataverse. This scenario occurs because no record in the BusinessEventsDefinition table has the combination of the affected table and the entity. Specifically, the LogisticsPostalAddress table isn’t the direct data source for the smmContactpersonCDSV2Entity entity. The smmContactpersonCDSV2Entity entity has smmContactPersonV2Entity as the data source, and smmContactPersonV2Entity, in turn, has LogisticsPostalAddressBaseEntity as the data source. The LogisticsPostalAddress table is the data source for LogisticsPostalAddressBaseEntity.

A similar situation can occur in some non-standard patterns, such as cases where the table that is being modified in Finance and Operations apps isn’t obviously linked to the entity that contains it. For example, the primary address data is computed on the smmContactPersonCDSV2Entity entity. The dual-write framework tries to determine how a change to an underlying table is mapped back to entities. Usually, this approach is sufficient. However, in some cases, the link is so complex that you must be specific. You must make sure that the RecId of the related table is directly available on the entity. Then add a static method to monitor the table for changes.

For an example, review the smmContactPersonCDSV2Entity::getEntityDataSourceToFieldMapping() method. CustCustomerV3entity and VendVendorV2Entity have been modified to handle this situation.

To fix the issue, follow these steps.

1. Add a PrimaryPostalAddressRecId field to the smmContactPersonV2Entity entity. Make it internal.

2. Add the same field to the smmContactPersonCDSV2Entity entity.
Error when you create a record where multiple records are sent from a Finance and Operations app to Dataverse in the same batch

For any transaction, a Finance and Operations app creates data in a batch and sends it as a batch to Dataverse. If two records are created as part of the same transaction, and they reference each other, you might receive an error message that resembles the following example in the Finance and Operations app:

Unable to write data to entity aaa_fundingsources. Un... Unable to lookup ebecsf_s_contracts with values {PC00...}. Unable to lookup aaa_fundingsources with values {PC00...}. Writes to aaa_fundingsources failed with error message: The remote server returned an error: (400) Bad Request.

To fix the issue, create entity relationships in the Finance and Operations app to indicate that the two entities are related to each other, and that the related records are handled in the same transaction.

Enable verbose logging of error messages

In a Finance and Operations app, you might encounter errors that are related to the Dataverse environment. The error message might not contain the full text of the message or other relevant data. To get more information, you can enable verbose logging by setting the IsDebugMode flag that is present on the DualWriteProjectConfigurationEntity entity in all project configurations in Finance and Operations apps.

1. Open the DualWriteProjectConfigurationEntity entity by using the Excel add-in. To use the add-in, enable design mode in the Finance and Operations Excel add-in, and add DualWriteProjectConfigurationEntity to a worksheet. For more information, see View and update entity data with Excel.
2. Set the IsDebugMode flag to Yes on the project.
3. Run the scenario.
4. The verbose logs are available in the DualWriteErrorLog table. To look up data by using the table browser, use the following URL: https://XXXaos.cloudax.dynamics.com/?mi=SysTableBrowser&tableName=DualWriteErrorLog
5. To capture more logs in debug mode, install the update in KB 4595434 (Fix for blank values being propagated in Dual write live sync).

Error when you add an address for a customer or contact

You might receive the following error message when you try to add an address for a customer or contact in Finance and Operations apps or Dataverse:

*Unable to write data to entity msdyn_partypostaladdresses.Writes to DirPartyPostalAddressLocationCDSEntity failed with error message Request failed with status code BadRequest and CDS error code : 0x80040265 response message: An error occurred in plugin. A record that has the attribute values Location ID already exists. The entity key Location ID Key requires that this set of attributes contains unique values. Select unique values and try again.*

To fix the issue, install the dual-write orchestration package version (2.2.2.60), so that the keys on the Address table are defined as shown in the following table.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Name</td>
<td>Location Key</td>
</tr>
<tr>
<td>Name</td>
<td>msdyn_locationkey</td>
</tr>
<tr>
<td>Fields</td>
<td>msdyn_locationid, parentid</td>
</tr>
<tr>
<td>Status</td>
<td>Active</td>
</tr>
<tr>
<td>System job</td>
<td>Blank</td>
</tr>
</tbody>
</table>

Error when you add a customer in Dataverse

You might receive the following error message when you try to add a customer in Dataverse:

"RecordError0":"Write failed for entity Customers V3 with unknown exception - Party record not found for party type 'Organization')."

When a customer is created in Dataverse, a new party number is generated. The error message is shown when the customer record, together with the party, is synced to Finance and Operations apps, but there is already a customer record that has a different party number.

To fix the issue, find the customer through party lookup. If the customer doesn’t exist, create a new customer record. If the customer does exist, use the existing party to create the new customer record.

Error when you create a new customer, vendor, or contact in Dataverse

You might receive the following error message when you try to create a new customer, vendor, or contact in Dataverse:

*Cannot update a party’s type from 'DirOrganization' to 'DirPerson', a delete of the existing party followed by an insert with the new type should be performed instead.*

In Dataverse, there is a number sequence on the msdyn_party table. When an account is created in Dataverse, a new party is created (for example Party-001 of the Organization type). This data is sent to the Finance and
Operations app. If the Dataverse environment is reset, or the Finance and Operations environment is linked to a different Dataverse environment, and then a new contact record is created in Dataverse, a new party value that starts with Party-001 is created. This time, the party record that is created will be Party-001 of the Person type. When this data is synced, Finance and Operations apps show the preceding error message, because party record Party-001 of the Organization type already exists.

To fix the issue, change the automatic number sequence for the msdyn_partynumber field of the msdyn_party table in Dataverse to a different automatic number sequence.

Performance issue with customer or contact mappings

You might be able to marginally improve the performance of live synchronization for customers and contacts by customizing the getEntityDataSourceToFieldMapping method (in the CustCustomerV3Entity entity) method and the getEntityDataSourceToFieldMapping method (in the smmContactPersonCDSV2Entity entity). These customizations reduce the number of records in the BusinessEventsDefinition table. This reduction in the number of records, in turn, reduces the number of events that are raised.

The getEntityDataSourceToFieldMapping method in the CustCustomerV3Entity entity makes sure that an update of the customer's electronic address or postal address triggers business events, so that the updated data will be sent to Dataverse. If you don't use all the fields and don't need the information in dual-write, comment out the appropriate lines in the method. Every tracked field and table that is added in this method adds a record in the BusinessEventsDefinition table for the combination of the tracked field and tracked entity.

```csharp
public static container getEntityDataSourceToFieldMapping(container mapping)
{
    mapping += [
        [tablestr(DirPartyBaseEntity), tablenum(LogisticsPostalAddress), fieldstr(CustCustomerV3Entity, AddressRecordId)],
        [identifierstr(DirPartyBaseEntity), tablenum(LogisticsElectronicAddress), fieldstr(CustCustomerV3Entity, PrimaryContactURLRecordId)],
        [identifierstr(DirPartyBaseEntity1), tablenum(LogisticsElectronicAddress), fieldstr(CustCustomerV3Entity, PrimaryContactPhoneRecordId)],
        [identifierstr(DirPartyBaseEntity2), tablenum(LogisticsElectronicAddress), fieldstr(CustCustomerV3Entity, PrimaryContactEmailRecordId)],
        [identifierstr(DirPartyBaseEntity3), tablenum(LogisticsElectronicAddress), fieldstr(CustCustomerV3Entity, PrimaryContactFaxRecordId)],
        [identifierstr(DirPartyBaseEntity4), tablenum(DirPartyLocation), fieldstr(CustCustomerV3Entity, DirPartyLocationRecordId)],
        [identifierstr(DirPartyBaseEntity5), tablenum(LogisticsPostalAddress), fieldstr(CustCustomerV3Entity, InvoiceAddressRecordId)],
        [identifierstr(DirPartyBaseEntity6), tablenum(LogisticsPostalAddress), fieldstr(CustCustomerV3Entity, DeliveryAddressRecordId)],
        [identifierstr(DirPartyBaseEntity7), tablenum(DirPartyTable), fieldstr(CustCustomerV3Entity, PartyRecordId)]];
    return mapping;
}
```

In a similar manner, the getEntityDataSourceToFieldMapping method in the smmContactPersonCDSV2Entity entity makes sure that any update of the contact's electronic address or postal address triggers business events, so that the updated data will be sent to Dataverse. In the method, you can comment out the lines for any fields that you don't use.
public static container getEntityDataSourceToFieldMapping(container mapping)
{
    mapping += [
        [tablestr(DirPartyBaseEntity), tablenum(logisticsPostalAddress),
            fieldstr(smmContactPersonCDSV2Entity, PrimaryPostalAddressRecId)],
        [identifierStr(DirPartyBaseEntity), tablenum(DirPartyTable), fieldstr(smmContactPersonCDSV2Entity, PrimaryAddressLocation)],
        [identifierStr(DirPartyBaseEntity1), tablenum(LogisticsElectronicAddress),
            fieldstr(smmContactPersonCDSV2Entity, PrimaryContactEmailRecordId)],
        [identifierStr(DirPartyBaseEntity2), tablenum(LogisticsElectronicAddress),
            fieldstr(smmContactPersonCDSV2Entity, PrimaryContactFaxRecordId)],
        [identifierStr(DirPartyBaseEntity3), tablenum(LogisticsElectronicAddress),
            fieldstr(smmContactPersonCDSV2Entity, PrimaryContactPhoneRecordId)],
        [identifierStr(DirPartyBaseEntity4), tablenum(LogisticsElectronicAddress),
            fieldstr(smmContactPersonCDSV2Entity, PrimaryContactFacebookRecordId)],
        [identifierStr(DirPartyBaseEntity5), tablenum(LogisticsElectronicAddress),
            fieldstr(smmContactPersonCDSV2Entity, PrimaryContactTwitterRecordId)],
        [identifierStr(DirPartyBaseEntity6), tablenum(LogisticsElectronicAddress),
            fieldstr(smmContactPersonCDSV2Entity, PrimaryContactURLRecordId)],
        [identifierStr(DirPartyBaseEntity7), tablenum(LogisticsElectronicAddress),
            fieldstr(smmContactPersonCDSV2Entity, PrimaryContactLinkedInRecordId)],
        [identifierStr(DirPartyBaseEntity8), tablenum(LogisticsElectronicAddress),
            fieldstr(smmContactPersonCDSV2Entity, PrimaryContactTelexRecordId)],
        [identifierStr(DirPartyBaseEntity9), tablenum(DirPartyTable), fieldstr(smmContactPersonCDSV2Entity, PartyRecordId)]];
    return mapping;
}

After you update the methods, follow these steps.

1. Sync the database, and build the application.
2. Stop all the dual-write maps on the smmContactPersonCDSV2Entity and CustCustomerV3Entity entities.
3. Start the maps. You should see fewer records in the smmContactPersonCDSV2Entity and CustCustomerV3Entity entities and the BusinessEventsDefinition table, and performance might marginally improve.
This topic provides troubleshooting information for dual-write integration between Finance and Operations apps and Dataverse. Specifically, it provides information that can help you fix issues with the Dual-write module in Finance and Operations apps.

**IMPORTANT**

Some of the issues that this topic addresses might require either the system admin role or Microsoft Azure Active Directory (Azure AD) tenant admin credentials. The section for each issue explains whether a specific role or credentials are required.

You can't load the dual-write module in a Finance and Operations app

If you can't open the Dual-write page by selecting the Dual Write tile in the Data management workspace, the data integration service is probably down. Create a support ticket to request a restart of the data integration service.

**Error when you try to create a new table map**

**Required credentials to fix the issue:** The same user that setup dual-write.

You might receive the following error message when you try to configure a new table for dual-write. The only user that can create a map is the user who setup the dual-write connection.

*Response status code does not indicate success: 401 (Unauthorized).*

**Error when you open the dual-write user interface**

You might receive the following error message when you try to access dual-write from the Data management workspace:

*login.microsoftonline.com refused to connect.*

To fix the issue, sign in by using an InPrivate window in Microsoft Edge, an incognito window in Chromium, or an incognito window in Google Chrome. You must also unblock or clear third-party cookies.

**Error when you link the environment for dual-write or add a new table mapping**

**Required role to fix the issue:** System administrator in both Finance and Operations apps and Dataverse.

You might encounter the following error when linking or creating maps:

*Response status code does not indicate success: 403 (tokenexchange).*

*Session ID: \{your session id\}*

*Root activity ID: \{your root activity\} id*
This error can occur if you don't have sufficient permissions to link dual-write or create maps. This error can also occur if the Dataverse environment was reset without unlinking dual-write. Any user with system administrator role in both Finance and Operations apps and Dataverse can link the environments. Only the user who setup the dual-write connection can add new table maps. After setup, any user with system administrator role can monitor the status and edit the mappings.

Error when you stop the table mapping

You might receive the following error message when you try to stop the table mappings:

```
[Forbidden], ["status":403,"source":null,"message":"Error from token exchange: User is not allowed to access connection dynamicscrmonline/xxxxxx-xxxx-xxxx-xxxxxxxx"], The remote server returned an error: (403) Forbidden.
```

This error occurs when the linked Dataverse environment isn't available.

To fix the issue, create a ticket for the Data Integration team. Attach the network trace so that the Data Integration team can mark the maps as Not running in the back end.

Errors while trying to start a table mapping

Unable to complete initial data sync

You might receive an error like the following when you try to run the initial data sync:

```
Unable to complete initial data sync. Error: dual-write failure - plugin registration failed: Unable to build dual-write lookup metadata. Error object reference not set to an instance of an object.
```

When you try to set that state of a mapping to Running, you might receive this error. The fix depends on the cause of the error:

- If the mapping has dependent mappings, then make sure to enable the dependent mappings of this table mapping.
- The mapping might be missing source or destination columns. If a column in the Finance and Operations app is missing, then follow the steps in the section Missing table columns issue on maps. If a column in Dataverse is missing, then click Refresh tables button on the mapping so that the columns are automatically populated back into the mapping.

Version mismatch error and upgrading dual-write solutions

You might receive the following error messages when you try to run the table mappings:

- Customer groups (msdyn_customergroups) : Dual write failure - Dynamics 365 for Sales solution 'Dynamics365Company' has version mismatch. Version: '2.0.2.10' Required version: '2.0.133'
- Dynamics 365 for Sales solution 'Dynamics365FinanceExtended' has version mismatch. Version: '1.0.0.0' Required version: '2.0.227'
- Dynamics 365 for Sales solution 'Dynamics365FinanceAndOperationsCommon' has version mismatch. Version: '1.0.0.0' Required version: '2.0.133'
- Dynamics 365 for Sales solution 'CurrencyExchangeRates' has version mismatch. Version: '1.0.0.0' Required version: '2.0.133'
- Dynamics 365 for Sales solution 'Dynamics365SupplyChainExtended' has version mismatch. Version: '1.0.0.0' Required version: '2.0.227'

To fix the issues, update the dual-write solutions in Dataverse. Make sure to upgrade to latest solution that matches the required solution version.
This topic provides troubleshooting information that can help you fix issues that are related to dual-write party and global address book functions.

**Verify these prerequisites**

Before you use the party and global address book functionality, make sure these are configured correctly.

- Integration keys.

<table>
<thead>
<tr>
<th>MAP</th>
<th>KEYS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account</td>
<td>accountnumber [Account Number]</td>
</tr>
<tr>
<td></td>
<td>msdyn_company.cdm_companycode [Company (Company Code)]</td>
</tr>
<tr>
<td>Contact</td>
<td>msdyn_contactpersonid [Account Number/Contact Person ID]</td>
</tr>
<tr>
<td></td>
<td>msdyn_company.cdm_companycode [Company (Company Code)]</td>
</tr>
<tr>
<td>Contact For Customer/Vendor</td>
<td>msdyn_contactforpartynumber [Contact For Party Number]</td>
</tr>
<tr>
<td></td>
<td>msdyn_associatedcompanyid.cdm_companycode [Associated Company (Company Code)]</td>
</tr>
<tr>
<td>Vendor</td>
<td>msdyn_vendoraccountnumber [Vendor Account Number]</td>
</tr>
<tr>
<td></td>
<td>msdyn_company.cdm_companycode [Company (Company Code)]</td>
</tr>
</tbody>
</table>

- Map versions. For more information, see [Party and global address book](#).

**Error about Location ID key when you try to add an address**

You might receive the following error message when you try to add an address to an account or contact in a Finance and Operations app or Microsoft Dataverse:

*Unable to write data to entity msdyn_partypostaladdresses. Writes to DirPartyPostalAddressLocationCDSEntity failed with error message Request failed with status code BadRequest and CDS error code: 0x80040265 response message: An error occurred in plugin. A record that has the attribute values Location ID already exists. The entity key Location ID Key requires that this set of attributes contains unique values. Select unique values and try again.*

To fix this issue, make sure that the key on the **Address** table is set as shown in the following table.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Name</td>
<td>Location Key</td>
</tr>
</tbody>
</table>
If Dual-write Party and Global Address Book Solutions is not installed, then the key on this table is set to the `msdyn_locationid` field. Install the dual-write orchestration solution version (version 2.2.2.60 or later). This replaces the previous key created on `Address` table.

### Error when you try to run Customers, Vendors, or Contacts V2 maps

You might receive the following error message when you try to run Customers, Vendors, or Contacts V2 maps:

**Customers V3 (accounts): Project validation failed. Missing destination field**

*msdyn_billingaccount.accountnumber in the schema.* **Missing destination field**

*msdyn_primarycontact.msdyn_contactforpartynumber in the schema.*

There are multiple keys defined on the `msdyn_company` table in Dataverse. Dual-write cannot determine which key to use as integration key, and it randomly assigns one of the keys as integration key. To fix this issue, update the integration keys manually as described in step 8 of *Party and global address book*. Then refresh the table mappings. The missing destination field error should disappear.

### Error that the Party ID is different between Finance and Operations apps and Dataverse

You might receive an error message that the Party ID is different between a Finance and Operations app and Dataverse for the Customers, Vendors, or Contacts V2 maps.

To fix this issue, use the latest version of maps as described in step 7 of *Party and global address book*.

### Errors when upgrading Dual-write Party and Global Address Book Solutions

You might receive error messages when you upgrade Dual-write Party and Global Address Book Solutions from 2.4.0155 to later versions.

The party and global address book functionality was part of the dual-write orchestration solution when it was released for preview in January and February 2021. Based on customer feedback, the functionality was released for General Availability as a separate solution. As a separate solution, the functionality is optional. If you are using the preview version of the dual-write orchestration solution that contains party and global address book functionality, then you need to uninstall the dual-write orchestration solutions or reset the Dataverse environment and get the latest solutions.

Dual-write Party and Global Address Book Solutions contains the following solutions.

- **Party** - Includes the schemas for party, postal address, and electronic address.
- **Dynamics365GABExtended** - Includes all code and schema changes to support Accounts, Vendors, Contacts, and Contact for party functionality. This support was separated from the
Error when you try to create a new contact from the View Contact form

You might receive the following error message when you try to create a new contract from the View Contact form in a Finance and Operations app:

*Unable to write data to entity msdyn_contactforparties. Unable to lookup msdyn_parties with values (000006057). Unable to lookup cdm_workers with values (000020).*

To fix this issue, create the Contact record using the Add Contact form.

Error when you try to update a contact

You might receive the following error message when you try to update a contact that originated in Dataverse in a Finance and Operations app.

*Unable to write data to entity msdyn_contactforparties.Writes to smmContactPersonV2Entity failed with error message Request failed with status code BadRequest and CDS error code: 0x0 response message: An error occurred while validating input parameters: Microsoft.OData.ODataException: Cannot convert the literal ’’ to the expected type ‘Edm.Int32’.*

To fix this issue, install the latest Dual-write Party and Global Address Book Solutions. This issue is fixed in version 3.0.0.26.

Error when you create a new customer, vendor, or contact in Dataverse

You might receive the following error message when you try to create a new customer, vendor, or contact in Dataverse:

*Cannot update a party’s type from ‘DirOrganization’ to ‘DirPerson’, a delete of the existing party followed by an insert with the new type should be performed instead.*

This issue occurs in non-production environments if users try connecting one Finance and Operations app to different Dataverse organizations, or if they try to reset the existing Dataverse organizations. The issue is due to the number sequence for Party ID in the msdyn_party table in Dataverse. The follow sequence of events generates the error:

1. An account is created in Dataverse. Dataverse creates a new party with Party ID **Party-001** and Party type **Organization**.
2. The new account is then sent to the Finance and Operations app.
3. The Dataverse environment is reset later or the same Finance and Operations apps environment is again connected to a different Dataverse organization.
4. You create a new contact this time in Dataverse. The number sequence for msdyn_party starts with **Party-001**. This time, the party record is created with **Party-001** and Party type as **Person**.
5. The data is synced to the Finance and Operations app. Because the Finance and Operations app already has **Party-001** as **Organization**, the error is generated.

To fix this issue, change the auto number sequence for the msdyn_partynumber field in the msdyn_party table to a different auto number sequence.
Error when you run the initial sync of party postal addresses and party electronic addresses

You might receive an error such as "the Party number could not be found" when you try to run the initial sync of party postal addresses and party electronic addresses.

There is a range added to the DirPartyCDSEntity entity in Finance and Operations apps to filter only parties of type Person and Organization. As a result, the initial sync of the CDS Parties – msdyn_parties mapping will not sync parties of other types, including Legal Entity and Operating Unit. When the initial sync runs for CDS Party postal addresses (msdyn_partypostaladdresses) or Party Contacts V3 (msdyn_partyelectronicaddresses) you might see errors, for example, that the Party number could not be found in Dataverse.

We are working to remove the party type range on the Finance and Operations apps entity so that parties of all types synchronize to Dataverse successfully. Check back to this topic for updates.
This topic provides troubleshooting information for dual-write integration between Finance and Operations apps and Dataverse. Specifically, it provides information that can help you fix issues that are related to solution awareness.

**IMPORTANT**

Some of the issues that this topic addresses might require either the system admin role or Microsoft Azure Active Directory (Azure AD) tenant admin credentials. The section for each issue explains whether a specific role or credentials are required.

**Error on the Dual-write page**

On the **Dual-write** page, you might receive an error message that resembles the following example:

*The entity with a name 'msdyn_dualwriteentitymap' with namemapping='Logical' was not found in the MetadataCache.*

To fix the issue, make sure that the dual-write core solution is installed in Dataverse. The dual-write core solution is a prerequisite for solution awareness.
This topic provides troubleshooting information for dual-write integration between Finance and Operations apps and Dataverse. Specifically, it provides information that can help you fix issues that are related to upgrades of Finance and Operations apps.

**IMPORTANT**

Some of the issues that this topic addresses might require either the system admin role or Microsoft Azure Active Directory (Azure AD) tenant admin credentials. The section for each issue explains whether a specific role or credentials are required.

### Database synchronization errors

**Required role to fix the issue:** System admin

You might receive an error message that resembles the following example when you try to use the `DualWriteProjectConfiguration` table to update a Finance and Operations app to Platform update 30.

```
Infolog diagnostic message: 'Cannot select a row in Dual write project sync (DualWriteProjectConfiguration).
```

To fix the issue, follow these steps.

1. Sign in to the virtual machine (VM) for the Finance and Operations app.
2. Open Visual Studio as an admin, and open the Application Object Tree (AOT).
3. Search for `DualWriteProjectConfiguration`.
4. In the AOT, right-click `DualWriteProjectConfiguration`, and select **Add to new project**. Select **OK** to create the new project that uses default options.
5. In Solution Explorer, right-click **Project properties**, and set **Synchronize Database on Build** to **True**.
6. Build the project, and confirm that the build is successful.
7. On the **Dynamics 365** menu, select **Synchronize database**.
8. Select **Synchronize** to do a full database synchronization.
9. After the full database synchronization is successful, rerun the database synchronization step in Microsoft Dynamics Lifecycle Services (LCS) and use the manual upgrade scripts as applicable, so that you can proceed
Missing table columns issue on maps

**Required role to fix the issue:** System admin

On the **Dual-write** page, you might receive an error message that resembles the following example:

*Missing source field <field name> in the schema.*

<table>
<thead>
<tr>
<th>Source field</th>
<th>Map type</th>
<th>Destination field</th>
<th>Issues</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>✓</td>
<td>msdyn_productname [Product Name]</td>
<td>Missing source field msdyn_productname in the schema...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>✓</td>
<td>msdyn_productnumber [Product Number]</td>
<td>Missing source field msdyn_productnumber in the schema...</td>
<td>...</td>
</tr>
</tbody>
</table>

To fix the issue, first follow these steps to make sure that the columns are in the table.

1. Sign in to the VM for the Finance and Operations app.
2. Go to **Workspaces > Data management**, select the **Framework parameters** tile, and then, on the **Table settings** tab, select **Refresh table list** to refresh the tables.
3. Go to **Workspaces > Data management**, select the **Data tables** tab, and make sure that the table is listed. If the table isn’t listed, sign in to the VM for the Finance and Operations app, and make sure the table is available.
4. Open the **Table mapping** page from the **Dual-write** page in the Finance and Operations app.
5. Select **Refresh table list** to automatically fill the columns in the table mappings.

If the issue still isn’t fixed, follow these steps.

**IMPORTANT**

These steps guide you through the process of deleting a table and then adding it again. To avoid issues, be sure to follow the steps exactly.

1. In the Finance and Operations app, go to **Workspaces > Data management**, and select the **Data tables** tile.
2. Find the table that is missing the attribute. Click **Modify target mapping** in the toolbar.
3. On the **Map staging to target** pane, click **Generate mapping**.
4. Open the **Table mapping** page from the **Dual-write** page in the Finance and Operations app.
5. If the attribute is not auto-populated on the map, add it manually by clicking **Add attribute** button and then clicking **Save**.
6. Select the map and click **Run**.
Verify dual-write configuration in Finance and Operations apps and Dataverse

This topic provides troubleshooting information for dual-write integration between Finance and Operations apps and Dataverse. Specifically, it explains how you can determine whether dual-write is configured in Finance and Operations apps and in Dataverse.

Verify that dual-write is configured in a Finance and Operations app

To determine whether the errors that you see when you try to save rows for update come from dual-write, first verify that dual-write is configured.

- If you have admin privileges in the Finance and Operations app, go to Workspaces > Data management, and select the Dual-write tile. If the details of the linked environments and the list of table maps that are running are shown, dual-write is configured.

<table>
<thead>
<tr>
<th>Table Map</th>
<th>Status</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales tax groups (msdyn_taxgroups)</td>
<td>Running</td>
<td>1.0.0.0 - Dynamics 365</td>
</tr>
<tr>
<td>Job detail (cdm_jobs)</td>
<td>Mappings created, not running</td>
<td>1.0.0.0 - Dynamics 365</td>
</tr>
<tr>
<td>Employment (cdm_employments)</td>
<td>Running</td>
<td>1.0.0.0 - Dynamics 365</td>
</tr>
<tr>
<td>Fiscal calendar year integration entity (msdyn_fiscalcalendaryears)</td>
<td>Running</td>
<td>1.0.0.0 - Dynamics 365</td>
</tr>
</tbody>
</table>

- If you don't have admin privileges, you will receive an error message, *Unable to write data to entity <entity name>*. In the example in the following illustration, you can't create a customer row in the Finance and Operations app, because dual-write is configured, but the customer group and payment terms reference data don't exist in Dataverse.

![Unable to write data to entity accounts](image)

For information about how to fix issues when you create data in Finance and Operations apps, see Troubleshoot live synchronization issues.

Verify that dual-write is configured in Dataverse

When you create data, if you see the **Company** column on pages in Dataverse, dual-write is configured.
For information about how to fix issues when you create data in Dataverse, see Troubleshoot live synchronization issues.

For information about how to view error details if you encounter any errors while you create data in Dataverse, see Enable and view the plug-in trace log in Dataverse to view error details.
This topic describes error codes for the table map health check.

**Error 100**

The error message is, "Minimum required Finance and Operations platform version is PU 43 to run Finance and Operations recommendations."

The feature requires platform updates for version 10.0.19 or later of Finance and Operations apps.

**Error 400**

The error message is, "No business events registration data found for the entity {Finance and Operations UniqueEntityName} which means either the map is not running or all the field mapping are unidirectional."

**Error 500**

The error message is, "No project configurations found for project {project name}. This could be either the project is not enabled or all the field mappings are unidirectional from customer engagement to Finance and Operations."

Check the mappings for the table map. If they are unidirectional from customer engagement apps to Finance and Operations apps, no traffic is generated for live synchronization from Finance and Operations apps to Dataverse.

**Error 900**

The error message is, "Invalid source filter {sourceFilter} format for entity {Finance and Operations UniqueEntityName}."

The source filter that is specified on the table map for Finance and Operations apps isn't syntactically correct. To validate the filter criteria, see Troubleshoot live synchronization issues.

**Error 1000**

The error message is, "Entity {Finance and Operations UniqueEntityName} query used for dual-write live sync is {Finance and Operations EntityFilterQueryString}. Records which meet the query criteria will be picked up for live sync."

The entity query that was returned is the backing SQL query for the entity. Check for inner joins or filters on the query that determine the business data that is being picked up for live synchronization. Inner joins and filters are mandatory conditions that must be fulfilled for each record that is being picked up for dual-write live synchronization.

**Error 1300**

The error message is, "Virtual fields {s.EntityFieldName} for entity {Finance and Operations EntityMetadata.EntityProperties.LogicalEntityName} may not be tracked for dual-write."

Virtual fields from Finance and Operations tables aren't enabled for tracking. Live synchronization can sync the
data, but it won't be able to pick up the changes that are made on the columns.

**Error 1500**

The error message is, “There should be at least one field mapped to a non lookup field on customer engagement to enable tracking on the data source {datasource.DataSourceName}.”

The data source from the entity doesn't have any field that is mapped for dual-write. Changes to the underlying table won't be tracked for dual-write.

**Error 1600**

The error message is, “Datasource: {datasource.DataSourceName} for entity (Finance and Operations EntityMetadata.EntityProperties.LogicalEntityName) has a range. Only records that satisfy the range condition are picked up for outbound.”

Entities in Finance and Operations apps can have data sources where filter ranges are enabled. These ranges define the records that are picked up as part of live synchronization. If some records are skipped from Finance and Operations apps to Dataverse, check whether the records meet the range criteria on the entity. A simple way to do this check is to run a SQL query that resembles the following example.

```sql
select * from <EntityName> where <filter criteria for the records> on SQL.
```

**Error 1700**

The error message is, “Table: {datasourceTable.Key.subscribedTableName} for entity {datasourceTable.Key.entityName} is tracked for entity {origTableToEntityMaps.EntityName}. Same tables tracked for multiple entities can impact system performance for live sync transactions.”

If the same table is tracked by multiple entities, any change to the table will trigger dual-write evaluation for the linked entities. Although the filter clauses will send only the valid records, the evaluation might cause a performance issue if there are long-running queries or unoptimized query plans. This issue might not be avoidable from the business perspective. However, if there are many intersecting tables across multiple entities, you should consider simplifying the entity or checking optimizations for entity queries.
This topic lists frequently asked questions about dual-write and provides brief answers to help you quickly get the information that you require.

**Dual-write setup**

**Do you plan to enable dual-write to use Dataverse as a hub between multiple Finance and Operations environments?** If Dataverse is used as a hub, data can be synced between two or more Finance and Operations environments.

The current plan of row is to restrict dual-write to a one-to-one (1:1) mapping between a single Finance and Operations environment and a single Dataverse environment.

**Can I control the sequencing of maps in dual-write, as I can in Data integrator?**

Dual-write is transaction-based. For example, if a change in a Finance and Operations app triggers synchronization of multiple maps with Dataverse, by default, those changes will be sequenced in the order in which they are updated in the database. This pattern makes more sense in the context of initial synchronization. The system provides related table maps in a specified order, and you can reorder the list so that it best suits your environment.

**Do application users require any special permissions to enable or configure dual-write?**

You must have two Azure Active Directory (Azure AD) applications set up for the Finance and Operations environment and two application users set up in the Dataverse environment. These application users should contain the appropriate application IDs. For the connection to work properly, you must give the applications the relevant table permissions by using a security role. For more information, see [Verify requirements and grant access](#).

**Do end users require any special permissions to enable or configure dual-write?**

End users who are configuring dual-write mappings should have System Administrator security roles assigned in both Dataverse and Finance and Operations environments.

Dual-write mappings can be accessed by multiple users, as long as all the users and environments belong to a single tenant, and the user has the required security and licenses assignment.

**I have multiple legal entities. Some of my maps are legal table–specific or valid for only some of the legal entities. What is the best way to address this requirement? Can I apply a filter such as Company = USMF to address it?**

Legal table mapping can be done when the Dataverse environment is linked. You can't map table maps to a specific legal entity.

**If dual-write solutions are installed in Dataverse, can I uninstall them?**

Dual-write solutions are managed solutions that can be uninstalled. However, when a managed solution is uninstalled, all components in the solution are deleted. Any data that is stored in the components is also deleted. For more information, see [Maintain managed solutions](#).

**I have data in both a customer engagement app and a Finance and Operations app, and I bootstrap my existing data in the customer engagement app. If my data isn't currently aligned, can I specify a master source for the initialization run, so that all differences are applied to the target?**

After the bootstrapping is done, you can configure the initial synchronization to apply differences and select a master. For more information about bootstrapping, see [Bootstrap with company data FAQ](#). For more
Dual-write administration and management

**What is the purpose of the integration key, and is it mandatory?**

The integration key is the natural key that uniquely identifies rows. Integration keys are required only for Dataverse tables. You can manually create an integration key in dual-write. An integration key can also be automatically created from the table’s alternate keys, if an alternate key is already provided for the table. Integration keys are used for the same purpose as alternate keys: they provide an efficient and accurate way to integrate data with external systems. Integration keys are essential in cases where an external system doesn’t store the globally unique identifiers (GUIDs) that uniquely identify rows in Dataverse.

Dual-write uses integration keys to uniquely identify rows, by using one or more table column values that represent a unique combination. For example, to identify an account row by using an integration key, you can use the account number column. Alternatively, you can use the account number column together with other columns that have values that should not change. For more information, see Define alternate keys using Power Apps portal.

It's important that keys be matched between the Finance and Operations environment and the Dataverse environment. Otherwise, issues might occur during the initial synchronization phase.

**How do I move table maps between environments? Is version control supported for table maps?**

You can export maps and then import them into a different environment. You can automate the process by using Azure DevOps. You can have version control on your dual-write mappings, because the mappings are solution-aware components. For more information, see Update table maps and export them to other environments as a solution.

**Where can I find examples and patterns for filtering dual-write maps?**

For basic filtering examples, see Filter your data.

For more advanced examples for Dataverse, see Filter results. Nested lookup isn't supported in dual-write source filters. Only standard filter operators directly against table columns are supported.

For more advanced Finance and Operations filters, see Using Expressions in Query Ranges and Advanced filtering and query syntax.

**Dual-write live synchronization introduces tight coupling across applications. What happens if one side fails? Will the other side fail too?**

When the integration is in live sync mode, if the sync fails on one of the apps, then the other app will fail as well and users will receive an error. When the integration is paused, changes are staged. They are then written when the target system is up and running. For more information about how to automatically pause integrations, see Alert notifications.

When live synchronization is paused and then resumed, does it follow the sequence of changes? For example, if the Name column in the Finance and Operations app is changed from NameA to NameB to NameC, is customer engagement data changed from NameA to NameB to NameC, or is it changed directly from NameA to NameC?

The integration follows the complete sequence of changes. In the example, the customer engagement app data is changed from NameA to NameB to NameC.

**How do I handle a Finance and Operations database transfer from PROD to STAGE? What is the effect on dual-write? After the transfer, the systems are no longer in sync. Is the synchronization done automatically?**

Each linked environment-pair (Finance and Operations apps environment and Dataverse environment) should be treated as a single unit and refreshed accordingly. For example, if you are refreshing a sandbox from production, then both Finance and Operations app sandbox environment and the Dataverse sandbox environment should be refreshed from their production counterparts. If dual-write is already used in target
environments, those environments need to be unlinked. After the data refresh on target environments, these tables should be cleaned up:

- Finance and Operations apps tables: `DualWriteProjectConfiguration`, `DualWriteProjectFieldConfiguration`, and `BusinessEventsDefinition`.
- Dataverse tables: `DualwriteRuntimeConfiguration`.

The environments need to be relinked and maps reactivated manually.

I need real-time integration, and I want to move some tables or scenarios from Data integrator to dual-write. How do I migrate, and what are the implications of changing my integration pattern?

For information about how to migrate Prospect to cash to dual-write, see Migrating data from Data Integrator to Dual Write. In general, three things might change during migration:

- Manual migration of the maps from Data integrator to dual-write
- Table changes, because of the absence of advanced query capabilities
- Data migration, because of adaptation to new concepts such as company striping

On Finance and Operations data tables, can I develop unbounded columns that flow to Dataverse by using dual-write?

Yes. You can use both computed columns and virtual columns. However, you should monitor the performance overhead from the additional X++ logic that is required for reads and writes. Round-tripping within the same transaction isn’t allowed. Therefore, you should avoid using virtual columns to transform or calculate additional values through X++ and expect that to go back to Dataverse within the same transaction.

When I use the Dataverse offline app, what happens if I can’t sync the data after reconnection? Does this situation cause an inconsistent state between the Dataverse environment and the Finance and Operations environment?

You can interact with Dataverse data offline when using the Dynamics 365 for phones app or the Field Service Mobile app in offline mode. In both apps, data is stored offline and can be synced with the server at your discretion. If there are errors when the offline data is synced with the server, and updates can't be done because the other environment is failing, data sync will fail, and Dataverse will not be updated. When the integration is paused, you can re-run the sync and save your updates on the server. These changes will be staged and then synced with the Finance and Operations environment when the mapping is up and running again. For more information, see Run model-driven apps and canvas apps on Power Apps mobile.

Mapping concepts between apps

How are number sequences handled? For example, the customer account number is automatically generated in Finance and Operations apps, but it’s added manually in customer engagement apps.

Number sequences for Finance and Operations apps and customer engagement apps aren’t connected. In a scenario that involves a multi-mastered table, you must either plan for separate number sequence formats or create a range for each app. Here are some examples:

- In the Finance and Operations app, use `F0001`, `F0002`, `F0003`. In the customer engagement app, use `C0001`, `C0002`, `C0003`.
- In the Finance and Operations app, use `US0001` to `US4999`. In the customer engagement app, use `US5000` to `US9999`.

If a table is created in only one system, set up the number sequence in the source app only. For more information, see Autonumber columns.

Can I map a company-specific table in a customer engagement app with a global table in a Finance and Operations app, or a global table in a customer engagement app with a company-specific table in a Finance and Operations app?

Dual-write supports mappings only between cross-company tables or company-specific tables from both sides.
How do I make a company-specific table in Dataverse?

You can make Dataverse custom tables company-specific by adding a many-to-one (N:1) relationship between your custom tables and the out-of-box company table. You should also include the company foreign key as part of the table key. For more information, see [Company concept in Dataverse](#).

To enable table maps for dual-write, you must define an alternate key in Dataverse. The value of the alternative key in Dataverse must match the key that is defined in the Finance and Operations app. For more information, see [Criteria for linking tables](#).

Can I merge records in customer engagement apps while using dual-write?

No, Finance and Operations apps do not permit the merging of records. Because of this, the merge functionality in customer engagement apps will not execute when a dual-write mapping is present on a table.

Is there a document about best practices for table usage? Should I use Customers V2, Customers V3, or Customer Details? What is the difference between these tables, and what is the use case for each?

You should use the [out-of-box scenarios](#) if you can, because they cover common scenarios such as customer/vendor integration.
Virtual entities overview

11/24/2021 • 4 minutes to read • Edit Online

**IMPORTANT**

This functionality requires version 10.0.12 of Finance and Operations apps, and service update 189 of Microsoft Dataverse. The release information for Dataverse is published on the latest version availability page.

Virtual entities for Finance and Operations apps

Finance and Operations apps are a virtual data source in Dataverse, and enable full create, read, update, and delete (CRUD) operations from Dataverse and Microsoft Power Platform. By definition, the data for virtual entities doesn't reside in Dataverse. Instead, it continues to reside in the app where it belongs. Before CRUD operations can be performed on Finance and Operations entities from Dataverse, the entities must be made available as virtual entities in Dataverse. CRUD operations can then be performed from Dataverse and Microsoft Power Platform on data that resides in Finance and Operations apps.

All Open Data Protocol (OData) entities in Finance and Operations apps are available as virtual entities in Dataverse, and therefore also in Microsoft Power Platform. Makers can now use data directly from Finance and Operations apps to build experiences in customer engagement apps. These experiences offer full CRUD capability and don't require copying to Dataverse. Power Apps portals can be used to build external-facing websites that enable collaboration scenarios for business processes in Finance and Operations apps.

Virtual entities for core Human Resources

Core Human Resources entities can also be virtualized, just as Finance and Operations entities can. For more information, see Configure Dataverse virtual tables.

Architecture

Virtual entities are a Dataverse concept that is useful beyond Finance and Operations apps. The following illustration shows how the Finance and Operations provider for virtual entities is implemented. The provider implements six primary methods. The first five methods are the standard CRUD operations: Create, Update, Delete, Retrieve, and RetrieveMultiple. The last method, PerformAction, is used to call OData actions, as described later in this topic. Calls to the Finance and Operations virtual entity data provider (shown as “virtual entities plug-in” in the illustration) will cause a Secure Sockets Layer (SSL)/Transport Layer Security (TLS) 1.2 secure web call to the CDSVirtualEntityService web API endpoint of Finance and Operations apps. This web service then converts the queries into calls to the associated physical entities in Finance and Operations apps, and invokes CRUD or OData operations on those entities. Because a Finance and Operations entity is directly invoked in all operations, any business logic on the entity or its backing tables is also invoked.
During calls, there are two points of translation from Dataverse to Finance and Operations apps. The first point of translation occurs in the VE Plugin, which translates concepts such as entity physical names into Finance and Operations entity names. It also converts some well-known concepts, such as Company references. The web service call still uses the EntityCollection, Entity, and QueryExpression objects to express the operations that are performed, by using the translated entity names and concepts from the VE Plugin. Finally, the CDSVirtualEntityAdapterService web API in Finance and Operations apps completes the translation from QueryExpression to QueryBuildDataSource and other internal Finance and Operations language constructs.

All calls between Dataverse and Finance and Operations apps as part of virtual entities are done as service-to-service (S2S) calls, by using the Azure Active Directory (Azure AD) application that is specified in the configuration. The user of this application should have access only to the CDSVirtualEntityAdapterService web API and the catalog entity, CDSVirtualEntityListEntity. These privileges are included in the out-of-box security role that is named CDSVirtualEntityApplication. During the S2S calls, Dataverse provides the identity of the user in Dataverse who is invoking the action. The CDSVirtualEntityAdapterService web API looks up the associated user in Finance and Operations apps and runs the query in the context of that user. Therefore, the S2S call doesn’t have to have explicit access to all the Finance and Operations entities. Instead, it can rely on the privileges of the user who is invoking the action to determine data access.

**NOTE**

We always recommend that you have both Finance and Operations apps and Dataverse co-located in the same Azure region, to ensure optimal latency in virtual entity calls. When Finance and Operations apps and Dataverse are co-located, the virtual entity overhead is expected to be less than 30 milliseconds (ms) per call.

Power Apps Portal can also access virtual entities. Because Power Apps Portal authorization is based on contact records, a mapping between contact records and Finance and Operations users is maintained in the dyn_externalportalusermapping table in Dataverse. This table should be editable only by highly privileged users in Dataverse who have the rights to control the security access that portal users have to Finance and Operations virtual entities. Any Finance and Operations user who is set up for Power Apps portal access must have the CDSVirtualEntityAuthorizedPortalUser security role assigned, and can’t have the system administrator or security administrator role assigned. Regardless of the Power Apps portal security setting that is applied to virtual entities, the resulting query to Finance and Operations apps is always run as the associated Finance and Operations user, and is subject to that user’s entity and row security settings. Anonymous portal access is also supported. For information about this type of access and how it can be done, see Power Apps Portal reference.
The public entity name that is exposed in Dataverse metadata for the Finance and Operations virtual entity uses the physical name of the Finance and Operations entity. This could be different from the public name of the entity as exposed by the OData metadata in Finance and Operations apps.

Building an app requires capabilities to perform relational modeling between entities that are being used in the app. In the context of virtual entities, there will be scenarios where virtual entities and native entities in Dataverse must work together to enable the desired user experience. This topic explains concepts of relational modeling that can be implemented using virtual entities for Finance and Operations.

Generating virtual entities

By default, virtual entities for Finance and Operations apps don’t exist in Dataverse. A user must query the catalog entity to view the entities that are available in the linked instance of Finance and Operations. From the catalog, the user can select one or more entities, and then request that Dataverse generate the virtual entities. This procedure is explained in later sections.

Entity fields

When a virtual entity is generated for a Finance and Operations entity, the system tries to create each field in the Finance and Operations entity in the corresponding virtual entity in Dataverse. In an ideal case, the total number of fields will be the same in both entities, unless there is a mismatch in supported data types between Finance and Operations and Dataverse. For data types that are supported, the field properties in Dataverse are set based on the properties in Finance and Operations.

This rest of this section describes supported and unsupported data types. For more information about fields in Dataverse, see Fields overview.

<table>
<thead>
<tr>
<th>DATA TYPE IN FINANCE AND OPERATIONS</th>
<th>MODELED DATA TYPE IN DATaverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td>Decimal</td>
</tr>
<tr>
<td></td>
<td>For information about the possible mismatch, see the next table.</td>
</tr>
<tr>
<td>Long</td>
<td>Decimal, where the precision equals 0 (zero)</td>
</tr>
<tr>
<td>Int</td>
<td>Integer</td>
</tr>
<tr>
<td>String (non-memo), String (memo)</td>
<td>String – single line of text, String – multiple lines of text</td>
</tr>
<tr>
<td>DATA TYPE IN FINANCE AND OPERATIONS</td>
<td>MODELED DATA TYPE IN DATaverse</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>UtcDateTime</td>
<td>DateTime (DateTimeFormat.DateTimeAndTime, DateTimeBehavior.TimeZoneIndependent)</td>
</tr>
<tr>
<td>An empty date (January 1, 1900) in Finance and Operations is surfaced as a null value in Dataverse.</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>DateTime - (DateTimeFormat.DateTimeOnly, DateTimeBehavior.TimeZoneIndependent)</td>
</tr>
<tr>
<td>An empty date (January 1, 1900) in Finance and Operations is surfaced as an empty value in Dataverse.</td>
<td></td>
</tr>
<tr>
<td>Enum</td>
<td>Picklist</td>
</tr>
<tr>
<td>Finance and Operations enumerations (enums) are generated as global OptionSets in Dataverse. Matching between the systems is done by using the External Name property of values. Enum integer values in Dataverse aren't guaranteed to be stable between the systems. Therefore, you should not rely on them, especially in the case of extensible enums in Finance and Operations, because these enums don't have a stable ID either. OptionSet metadata is updated when an entity that uses the OptionSet is updated.</td>
<td></td>
</tr>
</tbody>
</table>

Fields of the real and long data types in Finance and Operations are modeled as the decimal data type in Dataverse. Because of the mismatch in precision and scale between the two data types, the following behavior must be considered.

<table>
<thead>
<tr>
<th>USE CASE</th>
<th>RESULTING BEHAVIOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataverse has higher precision.</td>
<td>This use case should never occur unless the metadata is out of sync.</td>
</tr>
<tr>
<td>Finance and Operations has higher precision.</td>
<td>During a read operation, the value is rounded to the closest precision value in Dataverse. If the value is edited in Dataverse, it's rounded to the closest precision value in Finance and Operations. During a write operation, the value that is specified in Dataverse is written, because Finance and Operations supports higher precision.</td>
</tr>
<tr>
<td>Dataverse has higher scale.</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Finance and Operations has higher scale.</td>
<td>Dataverse shows the Finance and Operations value, even if it exceeds 100 billion. However, there will be a loss of precision. For example, 987,654,100,000,000,000 is shown in Dataverse as “987,654,099,999,999,900”. If the value of this field is edited in Dataverse, Dataverse validation throws an error that the value exceeds the maximum value before that value is sent to Finance and Operations.</td>
</tr>
</tbody>
</table>

The following data types in Finance and Operations aren’t supported in Dataverse. Fields of these data types in Finance and Operations entities won’t be made available in the corresponding virtual entities in Dataverse. If fields of these data types are used as parameters in Open Data Protocol (OData) actions, those actions won’t be available for use in the corresponding virtual entities. For more information about OData actions, see the OData actions section later in this topic.

- AnyType
Data types that are supported in Dataverse but not in Finance and Operations aren't supported in virtual entities for Finance and Operations.

**Entity key/primary key**

In Finance and Operations, entities can have one or more fields of various data types as the entity key. An entity key uniquely identifies a record in a Finance and Operations entity. Additionally, a record in an entity can be uniquely identified by a record ID primary key of the Int64 type.

In Dataverse, the primary key is always a globally unique identifier (GUID). The GUID-based primary key enables a record in an entity in Dataverse to be uniquely identified.

To bridge the implementation gap between Finance and Operations and Dataverse, the primary key of a virtual entity for Finance and Operations is a GUID (to comply with Dataverse). This GUID consists of the data entity ID in the first 4 bytes, and the record ID of the root data source in the entity as the last 8 bytes. This design satisfies Dataverse’s requirement that a GUID be used as the entity key. It also enables the table ID and record ID to be used to uniquely identify the entity record in Finance and Operations.

When using entities in Finance and Operations, you need to ensure that the root data source will always have a unique RecID. If this design is violated, duplicate GUID’s will show up in Dataverse for the corresponding virtual entity. Aggregate views are not supported via virtual entities for the same reason because these views may not have unique RecIDs.

**Primary field**

In Dataverse, each entity must have a primary field. This field must be a single field of the string type. The primary field is used in Dataverse in the following scenarios:

- The default views that are created for an entity include the primary field.
- The quick view form for an entity includes the primary field.
- A lookup to another entity is added to a page and shows the data from the primary field.

Based on this use of the primary field in Dataverse, the primary field for a virtual entity for Finance and Operations is designed to use the entity key of the corresponding entity in Finance and Operations.

Because the primary field in Dataverse is expected to have only one field of the string type, whereas the entity key in Finance and Operations can have multiple fields of various data types, the entity key fields are converted to strings. The strings are concatenated and separated by a pipe (|), to a maximum length of 255 characters. Any value that exceeds 255 is truncated. This virtual entity field that represents the primary field is named **mserp_primaryfield.**

**Relations**
Relations in Finance and Operations entities are modeled as one-to-many (1:n) or many-to-one (n:1) relations. These relations are modeled as relationships in the virtual entity in Dataverse. Note that many-to-many (n:n) relations aren’t supported in Finance and Operations.

For example, in Finance and Operations, if Entity A has a foreign key to Entity B, this relation will be modeled as an n:1 relationship in virtual entity Entity A in Dataverse. The schema name of this relationship in Dataverse uses the naming convention `mserp_FK_<source entity name>_<relation name>`. This naming convention has a maximum string length of 92 characters. Any relation where the schema name will produce a name that exceeds 92 characters won’t be generated in the virtual entity in Dataverse.

The external name of this relationship uses the naming convention `FK_<relation name>`. The external name is used to determine the relation in Finance and Operations when the query that is sent to Finance and Operations is built.

When a relationship is generated for a virtual entity in Dataverse, a new field of the lookup type is also added to the source entity. In the preceding example, when the relationship is created, a new lookup field that uses the naming convention `mserp_fk_<target_entity>_id` is added to source entity Entity A. Because there can be several relations in an entity in Finance and Operations, the same number of lookup fields (one per related entity) will be created in the source virtual entity. When this lookup field is added to a page or a view, it will show the primary field value from the related entity.

A relationship in the virtual entity in Dataverse will be generated only if the related entity in the relation already exists as a virtual entity in Dataverse. In the preceding example, if Entity B doesn’t exist as a virtual entity in Dataverse, the relation to Entity B won’t be created in Entity A when Entity A is generated as a virtual entity. This relation will be added to Entity A only when Entity B is generated as a virtual entity. Therefore, when a virtual entity is generated for Finance and Operations, validations are done to ensure that only relationships that can be complete and functional are generated in the virtual entity that is being generated.

In summary, a relationship to another Finance and Operations virtual entity might not exist in the virtual entity for either of the following reasons:

- The Finance and Operations entity that is participating in the relationship doesn’t exist as a virtual entity.
- The length of the name of the relationship exceeds 92 characters.

Note that if an error is encountered when any part of a Finance and Operations virtual entity is generated in Dataverse, the virtual entity won’t be created at all. If relationships don’t exist for either of the preceding reasons, the situation isn’t considered an error.

**Native entity–to–native entity relationships**

Native entity–to–native entity relationships are the standard Dataverse functionality, where relationships are resolved by using the GUID of the related entity. (This GUID is the entity key.) The GUID identifies the unique entity record in the related entity.

**Virtual table–to–virtual table relationships**

The relationships between two Finance and Operations virtual entities are driven by the relation metadata in the Finance and Operations entities. As was explained earlier, these relations are generated as relationships in Dataverse when the virtual entity is generated. As in the behavior for native entities in Dataverse, these relationships use the GUID to identify the unique record of the entity in Finance and Operations. Semantically, the GUID on the Finance and Operations virtual entity behaves like the GUID on the native Dataverse table. For information about the implementation of the GUID in Finance and Operations virtual entities, see the Entity
Virtual table–to–native table relationship

As was explained earlier, the GUID is the only information that is used to uniquely identify a record in a native Dataverse table (including in native entity–to–native entity relationships) or in a Finance and Operations virtual entity (including in virtual entity–to–virtual entity relationships). However, consider an example where you want to show sales orders from Finance and Operations for Account A in Dataverse. The query that is sent to Finance and Operations for this relationship will have a WHERE clause on the GUID of the entity key of the native accounts entity in Dataverse, because the sales orders must be filtered for a specific account in Dataverse. However, because Finance and Operations doesn’t have any information about the GUID of the entity in Dataverse, the query won’t return any sales orders. The query will be successful only if the WHERE clause has conditions that are based on the fields that Finance and Operations understands.

Therefore, how can the GUID of the accounts entity in Dataverse be replaced with fields that are in Finance and Operations, in such a way that the query that is sent to Finance and Operations will return the correct list of sales orders?

To solve this issue and enable a rich set of scenarios that allows for virtual entity–to–native entity relationships, relationships can be added to this type of entity. The relation will appear as a relationship when the virtual entity is synced.

In the above example, the relationship between the SalesOrderHeader virtual entity and the Account native entity should be based on the Account Number and Company fields. By default, the native account entity in Dataverse does not have a company field. For this example, we will add a company lookup field named new_testcompany to the native Account entity.

Next, we add a new key named new_accountcompanyidx, which specifies that (accountnumber, new_testcompany) together represent a unique row in the account entity in Dataverse.

The next step is to define this relationship in X++. The following example shows sample X++ code. The names of the fields, index, and mapping information should match the names of the fields and indexes created in Dataverse. In this example, a relationship named “synthaccount” will be created between the virtual SalesorderHeader entity and the native account entity in Dataverse. The mapped fields make up the new_accountcompanyidx index. The display name for the relationship will be @SYS11307. Note the backslash at the start of the display name. This ensures that the label defines the relationship, so that it is appropriately translated.

The field mapping indicates which field on the virtual entity maps to the field on the native entity. In the field mapping, the key is the virtual entity field, and the value is the native entity field.
The next step is to generate or refresh the virtual entity to get the new relationship. Note that relationships between a virtual entity and a native entity cannot be updated in Dataverse once it is created. The only way to make an update is to physically remove the relationship, refresh the entity, and then physically re-add the relationship in order to resolve the issue.

This relationship looks like a typical GUID-based relationship, but has extra metadata to translate query filters on the relationship into restrictions on the backing fields. The query that is now generated will have a WHERE clause that is based on the fields that Finance and Operations apps recognize. That query will then return the filtered list of sales orders, as expected.

Native entity–to–virtual entity relationships

Native entity–to–virtual entity relationships works much like native entity–to–native entity relationships. Users associate native records with virtual records in Finance and Operations, and the GUID of the virtual entity is saved on the native entity record. As was explained earlier, the entities that participate in a relationship will have the GUID field of the related entity on them. Therefore, when a quotation in Dataverse is associated with a customer in a Finance and Operations virtual entity, the GUID of the customer virtual entity will be saved in the quotation entity. This behavior enables records to be retrieved as expected, by using standard Dataverse functionality.

Enums

 Enums in Finance and Operations are modeled as OptionSets in Dataverse. When a virtual entity for Finance and Operations is generated, the required enums are generated as OptionSets. If an OptionSet already exists, it's used instead.

Company

An entity in Finance and Operations can be bound to a company, or it can be global. The virtual entity for a Finance and Operations entity that is bound to a company will have a relationship to the cdm_company entity in Dataverse. The cdm_company entity is a native entity in Dataverse and is part of the Dynamics365Company solution. As always, when a relationship is created, a lookup field is also created in the virtual entity for the related entity (cdm_company in this case). This lookup field is named Company, and it must be used to provide an optimal user experience where users can select a value in a list or go to the details of the related record. A field that is named Company Code is also added in the virtual entity. The value is a four-character string. This field must be used in programming.

Attachments

Attachments in Finance and Operations entities are supported on a per-entity basis. For example, an invoice header entity will implement an invoice-related attachments entity to enable attachments via entities.

Entities of this type will have relations with the corresponding attachments entity in Finance and Operations. Therefore, they will follow the same pattern as the other relations that were discussed earlier. In other words,
Finance and Operations entities that have implemented attachments functionality will also make attachments available by using virtual entities. Finance and Operations entities that don't support attachments also won't support attachments when they are virtualized in Dataverse.

Note that Finance and Operations virtual entities support only the reading of attachments. They don't currently support the creation, update, or deletion of attachments by using virtual entities.

**OData actions**

OData actions in the Finance and Operations entities are made available as custom actions in Dataverse. For more information about custom actions and what they enable in Dataverse, see [Custom actions](#).

Input and output parameters of the following types are supported. If an input or output parameter is of a different type, the OData action doesn't appear as the SDK message in Dataverse.

- Integer
- String
- Guid
- Boolean
- Date/Datetime

Here are some examples of OData actions that are supported in Finance and Operations entities, but that aren't supported in the corresponding virtual entities in Dataverse:

- RetailStoreTenderTypeTable.queryDistinctTenderTypeIdAndName (a collection of RetailStoreTenderTypeTable entity)
- DocumentRoutingClientApp.syncPrinters (DocumentRoutingClientApp entity)
- DocumentRoutingClientApp.updateJobStatus (DocumentRoutingJobStatus enum)
- DimensionCombination.getCombinationDisplayValue (LedgerJournalACType enum)

**Labels and localization**

Labels that are defined on metadata, such as entity names and field names in Finance and Operations, are retrieved when virtual entities are generated in Dataverse. The labels are retrieved by passing the list of language locales that are installed in Dataverse. Finance and Operations returns each label as a list of locale/value sets that are then used to construct a label instance in Dataverse. Only the language packs that exist at the time of entity generation or update are included. Additionally, only labels that Finance and Operations has provided a translation for are included. Any missing translations revert to the label ID, such as @SYS:DataEntity. After a new language pack is installed in Dataverse, existing entities must be updated to pick up the new label information, if labels in that language exist in Finance and Operations.

Any runtime labels are returned in the language of the current user context. In other words, they are returned in the language that is specified on that user's UserInfo record in Finance and Operations. This behavior also applies to error messages.

**Error handling**

Finance and Operations create, read, update, and delete (CRUD) business logic on entities and backing tables is run when it's called through the virtual entity in Dataverse. If any exception is thrown on the Finance and Operations side, the last message in the error log is returned to Dataverse and is thrown as an InvalidPluginExecutionException exception that contains the message from Finance and Operations. Because the Finance and Operations code runs in the context of the user, the language of the error message is based on the language that is specified on the UserInfo record in Finance and Operations. If any messages that are written to the info log in Finance and Operations don't result in an exception, they aren't shown in Dataverse.
Calculated/unmapped fields

Calculated and unmapped fields in Finance and Operations entities are also available in the corresponding virtual entities in Dataverse.
The application lifecycle for an end-to-end solution using Finance and Operations virtual entities will encompass both Finance and Operations as well as Dataverse. This topic explains this in detail.

Solution management

Virtual entities for Finance and Operations don't exist in Dataverse until they are created. Virtual entities must be created inside a solution. The MicrosoftOperationsERPVE solution is used for this purpose. This solution will contain all the virtual entities that are created from an instance of Finance and Operations.

MicrosoftOperationsERPVE is a managed solution. By definition, a managed solution can't be modified after it has been generated. However, MicrosoftOperationsERPVE is a managed solution that grants privileges to update the components (that is, virtual entities) that are inside it. Therefore, new virtual entities can be added to the solution as they are created, and existing virtual entities can be updated as required. Nevertheless, the privileges to modify the managed solution are available only to the platform itself. Users can't make changes directly to the solution.

Because MicrosoftOperationsERPVE is a managed solution, solutions from customers, partners, and independent software vendors (ISVs) can take a dependency on it. This capability allows for consistent application lifecycle management (ALM) for solutions that use and depend on the virtual entities for Finance and Operations.

When a solution that depends on MicrosoftOperationsERPVE is exported, placeholders for the virtual entities that are used in the solution are added in the exported solution. When that solution is imported into another Dataverse environment, the import process also generates the dependent Finance and Operations virtual entities in the MicrosoftOperationsERPVE solution for the Finance and Operations instance that is connected to the Dataverse environment. Therefore, MicrosoftOperationsERPVE must already exist before a solution that depends on it is imported. Otherwise, an error message is shown. Additionally, if a dependent entity isn't available in the Finance and Operations instance, the virtual entity for that entity won't be generated. Virtual entities are generated only for entities that are available.

NOTE

If a virtual entity already exists in Dataverse and a solution is being imported that now references new fields in the virtual entity which does not exist in Dataverse, a manual refresh must be performed on the virtual entity to get the latest metadata from Finance and Operations.

The following list describes other solutions that Finance and Operations virtual entities require to work, and that must be available in the Dataverse environment:

- MicrosoftOperationsERPCatalog – This solution provides a catalog of the available entities in a Finance
and Operations instance. It also provides the connection that is used to set up a configuration. For more information, see the later sections of this topic.

- **MicrosoftOperationsVESupport** – This solution provides the virtual entity provider for Finance and Operations apps. The provider can communicate with Finance and Operations apps and Dataverse. For more information, see the next section.

- **Dynamics365Company** – This solution adds the Company entity, which is referenced by all Finance and Operations entities that have a `PrimaryCompanyContext` metadata value.

All these solutions must be present in an environment. Otherwise, virtual entities won't work with Finance and Operations apps. These solutions are packaged together to allow for easier portability across environments.

## Managing entities from multiple environments

The MicrosoftOperationsVESSupport solution consists of the `msdyn_financeandoperationsvirtualentity` entity. This entity represents the virtual entity data source for Finance and Operations that captures connection setup information. Each record in this entity represents a connection to a Finance and Operations instance.

A catalog is used to list all the entities in a Finance and Operations instance that are available for virtualization in Dataverse (in other words, all the entities in Finance and Operations that are enabled for Open Data Protocol [OData]). The catalog is part of the default MicrosoftOperationsERPCatalog solution and is applicable to a Finance and Operations instance.

Note that each Dataverse environment must point to only one Finance and Operations instance at any time, and each Finance and Operations environment must point to only one Dataverse environment. Therefore, there should be only one record in the `msdyn_financeandoperationsvirtualentity` entity.

The `mserp_financeandoperationsentity` entity that represents the catalog can be queried to list the entities in a Finance and Operations instance. Because this entity is a virtual entity, the catalog is never persisted in Dataverse.

Notice that the name of the catalog entity has the "mserp_" prefix. This prefix identifies the entities in the catalog as Finance and Operations entities. The same prefix is also added to the system names of the virtual entities that are generated for Finance and Operations in the MicrosoftOperationsERPVE solution. Therefore, the maker can distinguish Finance and Operations virtual entities from other entities. The prefix is set in the managed solution and can't be changed.

## Managing entities from multiple ISV solutions

One or more ISV solutions will take a dependency on the MicrosoftOperationsERPVE solution to use virtual entities for Finance and Operations. Because custom entities in Finance and Operations use the same catalog as out-of-box entities in Finance and Operations, the virtual entities for custom Finance and Operations entities will also be generated in the MicrosoftOperationsERPVE solution.

The established guidelines and ALM for entity development in Finance and Operations ensure that there are no conflicting entity names across ISV solutions. Therefore, no conflicts of this type can occur when virtual entities are generated in Dataverse for custom Finance and Operations entities from multiple ISV solutions. All virtual entities for Finance and Operations entities, including custom entities, will have the "mserp_" prefix that was mentioned earlier.

## Managing a Finance and Operation instance in a Dataverse environment for virtual entities

One Finance and Operations instance must be linked to a Dataverse environment for virtual entities. The connection setup information that is required is captured in a virtual entity data source for Finance and Operations. This data source is included in the MicrosoftOperationsERPCatalog solution.
This topic explains how to configure virtual entities for Finance and Operations apps in Microsoft Dataverse.

**IMPORTANT**

The configuration steps in this topic are required only for Finance and Operations apps environments for which the Microsoft Power Platform integration is *not* enabled. For Finance and Operations apps environments for which the Microsoft Power Platform integration is enabled, the virtual entity configuration that is outlined in this topic is automatically done as part of the process for enabling the integration. For more information about how to enable the Microsoft Power Platform integration for Finance and Operations apps environments, see [Enable the Microsoft Power Platform integration](#).

The Dataverse solution for Finance and Operations virtual entities must be installed from Microsoft AppSource. For more information, see [Finance and Operations virtual entity](#).

Ensure the following solutions are installed in Dataverse.

- **Dynamics365Company** - This adds the Company entity, which is referenced by all Finance and Operations entities with a PrimaryCompanyContext metadata value.
- **MicrosoftOperationsVESupport** - This provides the core support for the Finance and Operations virtual entity feature.
- **MicrosoftOperationsERPCatalog** - This provides a list of available Finance and Operations entities through the mserp_financeandoperationsentity virtual entity.
- **MicrosoftOperationsERPVE** - This is the API-managed solution, which will contain the generated virtual entities as they are made visible.

When updates are available for the virtual entity solution, they can be manually applied in the Power Platform admin center. For more information about how to manually install and update the virtual entity solution, see [Manage Dynamics 365 apps](#).

**NOTE**

For Finance and Operations apps environments that the Microsoft Power Platform integration is enabled for, available updates to the virtual entity solution are automatically applied.

**Authentication and authorization**

After the solutions are imported into the Dataverse environment, both environments must be set up to connect to each other. Dataverse will call Finance and Operations apps by using Service-to-Service (S2S) authentication, based on an Azure Active Directory (Azure AD) application. This new Azure AD application represents the single instance of the Dataverse environment. If you have multiple pairs of Dataverse and Finance and Operations apps environments, separate Azure AD applications must be created for each pair to ensure that connections are established between the correct pair of Finance and Operations apps and Microsoft Power Platform environments.
Register the app in the Azure portal
The following procedure explains how to create the Azure AD application.

**IMPORTANT**
The Azure AD application must be created on the same tenant as the Finance and Operations apps.

2. Select New Registration. Enter the following information:
   - **Name** - Enter a unique name.
   - **Account type** - Enter *Any Azure AD directory* (single or multi-tenant).
   - **Redirect URI** - Leave blank.
   - Select Register.
   - Make a note of the Application (client) ID value, because you will need it later.
3. Create a symmetric key for the application.
   - Select Certificates & secrets in the newly created application.
   - Select New client secret.
   - Provide a description and an expiration date.
   - Select Save. A key will be created and displayed. Copy this value for later use.

Grant app permissions in Finance and Operations apps
The Azure AD application that you created will be used by Dataverse to call Finance and Operations apps. Therefore, it must be trusted by Finance and Operations apps and associated with a user account that has the appropriate rights. A special service user that has rights only to the virtual entity functionality must be created in Finance and Operations apps. This service user must have no other rights. After you complete this step, any application that has the secret of the Azure AD application that you created will be able to call this Finance and Operations apps environment and access the virtual entity functionality.

1. In Finance and Operations, go to System Administration > Users > Users.
2. Select New to add a new user. Enter the following information:
   - **User ID** - Enter dataverseintegration (or a different value).
   - **User name** - Enter dataverse integration (or a different value).
   - **Provider** - Set to NonAAD.
   - **Email** - Enter dataverseintegration (or a different value, does not need to be a valid email account).
   - Assign the security role CDS virtual entity application to this user.
   - Remove all other roles including System user.
3. Go to System Administration > Setup > Azure Active Directory applications to register Dataverse.
   - Add a new row.
   - **Client ID** - The Application (client) ID created above
Configure the virtual entity data source

The next step in the process is to provide Dataverse with the Finance and Operations instance to connect to. The following steps walk through this part of the process.

1. In Dataverse, go to Advanced Settings > Administration > Virtual Entity Data Sources.
2. Select the data source named “Finance and Operations”.
3. Fill in the information from the steps above.
   - **Target URL** - The URL at which you can access Finance and Operations.
   - **OAuth URL** - https://login.windows.net/
   - **Tenant ID** - Your tenant, such as “contoso.com”.
   - **AAD Application ID** - The Application (client) ID created above.
   - **AAD Application Secret** - The secret generated above.
   - **AAD Resource** - Enter 00000015-0000-0000-c000-000000000000 (this is the Azure AD application representing Finance and Operations, and should always be this same value).
4. Save the changes.

When the virtual entity configuration is completed, you can enable the virtual entities in Dataverse. For more information, see Enable Microsoft Dataverse virtual entities.
Because many entities that are available in Finance and Operations apps are enabled for Open Data Protocol (OData), the entities aren’t available as virtual entities in Microsoft Dataverse by default. After configuration is completed for Finance and Operations apps virtual entities in Dataverse, the virtual entities can be enabled in the Dataverse environment. Administrators can then determine which entities will be exposed as virtual entities that can be used in Dataverse.

**NOTE**
Configuration of Finance and Operations apps virtual entities in Dataverse is a prerequisite for enabling the virtual entities. The configuration is automatically done for Finance and Operations apps environments that are linked to a Microsoft Power Platform environment. For unlinked environments, manual configuration must be completed before the virtual entities can be enabled. For step-by-step information about how to configure Finance and Operations apps virtual entities in Dataverse, see Configure Dataverse virtual entities.

Generate virtual entities

1. In Dataverse, select the **Advanced find** filter button.

2. Search for **Available Finance and Operations Entities**, and select **Results**.

3. Find and open the entity that you want to enable.

4. Select the **Visible** checkbox, and then save your change.
The virtual entity is generated and will appear on all the appropriate menus. For example, it will appear in the **Advanced find** dialog box.

**Refresh virtual entity metadata**

You can force a refresh of a virtual entity’s metadata when you expect that the entity metadata in Finance and Operations apps has changed. To force a refresh, select the **Refresh** checkbox, and then save your change. The **latest entity definition from Finance and Operations apps is synchronized to Dataverse,** and the virtual entity is updated.

**Reference virtual entities**

All virtual entities are generated in the **MicrosoftOperationsERPVE** solution, which is API managed. Items in the solution change as you make entities visible or hidden. Nevertheless, the solution is still a managed solution that you can take dependencies on.

The standard Application Lifecycle Management (ALM) flow is to take a standard reference to a virtual entity from this solution. You can do this by using the **Add existing** action in the independent software vendor (ISV) solution. The virtual entity will be shown as a missing dependency of the solution, and it will be checked when the solution is imported. During import, if a specified virtual entity doesn't exist, it's automatically made visible. No additional work is required.

Follow these steps to consume virtual entities:

1. In Dataverse, follow the usual steps to create a separate solution that contains the consuming logic. See **Create a solution** in the Power Platform documentation for additional information on creating a solution.
2. Select **Add existing > Table**.
3. In the **Add existing tables** list, select the virtual entity that you want to reference.

4. When you're prompted to select assets to add, select any forms, views, or other elements that you want to customize, and then select **Finish**.

You can use the development tools to modify existing elements for the virtual entity, such as forms. You can also add new forms, views, and other elements.

When the solution is exported, it will contain hard dependencies on the virtual entity that is generated in the **MicrosoftOperationsERPVE** solution.
Power Apps portals will enable create, update, and delete (CRUD) operations to Finance and Operations entities that are available as virtual entities in Dataverse. This topic explains the scenarios that are implemented in Power Apps portals for Finance and Operations apps.

Anonymous access from Power Apps portals

Collaboration scenarios in business processes such as bidding or onboarding of prospects in Finance and Operations require that external users participate from the Power Apps portal, even though they aren't users in Finance and Operations apps. The simplicity of anonymous access is appealing in these types of scenarios because the users, who might not be Finance and Operations apps users, don't have to sign in. However, they are expected to perform CRUD operations in Finance and Operations to complete any meaningful tasks in the business processes.

To ensure that only the required entities are enabled for anonymous access, a user in Finance and Operations must be designated as the user who is used for anonymous access. This designation is configured in the Anonymous portal access user ID field on the Virtual entity tab on the System parameters page (System administration > System parameters). The designated user can then be assigned to duties and security roles to control access to specific data that must be made available to all users who will interact anonymously from the Power Portal.

Note that because this scenario involves anonymous access, the only user context that matters, from a security perspective, is the user who is designated in the Anonymous portal access user ID field.

Authenticated access from Power Apps portals

Fully authenticated user access from Power Apps portals to Finance and Operations lets users in Finance and Operations also interact from Power Apps portals. A user who signs in to the Power Apps portal is also a known user in Finance and Operations who has appropriate security roles based on job requirements. These roles govern the security access to data for the authenticated user in Power Portal. In addition, any Finance and Operations user that is expected to also use Power Apps portal to access Finance and Operations data must also belong to the CDSVirtualEntityAuthenticatedPortalUser security role. This provides an additional layer of security and also provides a way to know the total users that are authorized to access from Power Apps portals.

Because Power Apps portals authentication is linked to the Contacts entity in Dataverse, a mapping must be established between the Dataverse contact and the corresponding user in Finance and Operations. This mapping can be done by adding entries to the msdyn_externalportalusermapping entity. From a security perspective, the scope of virtual entities that are made available to authenticated users must be configured as Global in the Power Apps portal.

When authenticated users from a different tenant need to be added to Finance and Operations as users, you must use the Create new user process in Finance and Operations. This process adds cross-tenant users as Microsoft Azure Active Directory (Azure AD) business-to-business (B2B) guest users.
NOTE
Access from the Power Apps Portal will fail if the user (authenticated or anonymous) has been assigned the System administrator role in any Finance and Operations apps.
This topic is a collection of frequently asked questions about Finance and Operations virtual entities.

Do Tier 1 Finance and Operations environments or demo topologies work?
Yes, Tier 1 and DEVTEST and DEMO topologies should work.

What version of Finance and Operations do I need?
10.0.12 is the minimum version that is required.

Can a solution from an independent software vendor (ISV) take a dependency on virtual entities? What does the application lifecycle management (ALM) look like?
Yes. The virtual entities are all generated in the MicrosoftOperationsERPVE solution, which is API-managed. In other words, the items in the solution change as you make entities visible or hidden, but the solution is still a managed solution that you can take dependency on. The standard ALM flow just takes a standard reference to a virtual entity from this solution with the Add existing option in the ISV solution. Missing dependency of the solution will be checked when the solution is imported and during import, if a specified virtual entity doesn’t yet exist, the virtual entity is automatically made visible.

Which entities from Finance and Operations do users see in the catalog in Dataverse?
Generally, users see all entities where IsPublic is set to Yes. These entities are the same entities that are currently visible in Open Data Protocol (OData).

Do all Microsoft Power Platform users have to be users in Finance and Operations?
Any interactive user of Microsoft Power Platform who tries to access Finance and Operations data through a virtual entity must also exist as a user in Finance and Operations. Therefore, technically, not all users have to be users in Finance and Operations. Only those users who access Finance and Operations data through virtual entities must be users in Finance and Operations.

A S2S application user can also be used to call into virtual entities. For this kind of integration, the application user must be set up in System administration > Setup > Configure Azure Active Directory Applications. This allows for applications to integrate with Finance and Operations using virtual entities.

Where do I find the catalog entity?
In the Advanced find window, the entity is named Available Finance and Operations Entities.

Is there a way to specify a company when I perform data operations on a virtual entity?
Yes. Although the company is implicit in Finance and Operations, it's an explicit field on each company-striped entity in Dataverse. You can use either the Company Code field, where the value is a four-character string, or the Company field, which is a lookup to cdm_Company. Both approaches provide the same information.

Can I change the prefix for the virtual entities?
No. All Finance and Operations virtual entities should be generated in the MicrosoftOperationsERPVE solution, and they should all have the "mserp_" prefix. This prefix should not be changed. If you have a scenario where
you believe the prefix has to be changed, you should share that scenario with Microsoft.

**How can I filter data in an app that is created by using Power Apps, based on the current user or any other dynamic criteria, such as today-10?**

You can write a pre-operation plug-in on the RetrieveMultiple message of the entity and change the criteria on the query in it. Alternatively, you can write a post-operation plug-in to filter the results before they are returned.

**Can I pin a model-driven app into Finance and Operations?**

No, it isn’t currently possible to pin a model-driven app into Finance and Operations.

**How can I show, in the same grid, data from multiple virtual entities that are joined to a physical entity record in Dataverse?**

This approach isn’t currently possible in Dataverse.

**How do I add subcomponents in the new Power Apps experience?**

As in the previous Power Apps user interface (UI), you must redo the **Add Existing** operation. After the solution is selected, and Customer Groups has already been added as an entity, follow these steps.

1. Select **Add existing** > **Entity**.
2. Select customer group entity, and then select **Next**.
3. Under **Components**, select **Select components**.
4. Select the fields, relationships, and forms that you want, and then select **Add**.

**If I want a default value to be entered in a field during pre-create, will an initValue on the data entity work?**

Yes. Here is the order of calls:

1. Dataverse sends a create or update message.
2. All the existing logic on the Finance and Operations entity and backing tables is invoked. This logic includes default value entry that might change values.
3. Dataverse sends another Retrieve (single) message to get the latest copy of the data, including any fields that default values were entered for.

**Can I debug Finance and Operations when we do a create, read, update, and delete (CRUD) operation from Dataverse? If so, which process do I have to attach?**

Yes, to debug in Finance and Operations, open Visual Studio as an admin. Typically, Finance and Operations apps run under w3wp.exe as a process. However, when you open Visual Studio as an admin, IISExpress.exe is automatically opened, and Finance and Operations is hosted there. You can attach to IISExpress.exe (or to w3wp.exe if not running Visual Studio as an admin). To set breakpoints in the virtual entity code, find the **CDSVirtualEntityAdapter** and **CDSVirtualEntityController** classes. The adapter class is the first class that is called, and it only does serialization/deserialization. It then delegates to the controller class to do the actual queries. Therefore, the controller class is usually the easiest place to put breakpoints.

**Does the form business logic in Finance and Operations get called through virtual entities?**

Finance and Operations business logic that resides on forms isn’t invoked through virtual entities. Instead, you should expect the same behavior that you get through OData access to the same entities. The expectation is that an entity that is exposed to OData (that is, **IsPublic** is set to **Yes**) has appropriate protections to ensure that data can’t be corrupted. If any entity lacks this protection, that situation represents a bug in the entity. If you see differences in entity behavior between OData and virtual entities, that situation represents a bug in the virtual entity feature.

**If I develop a new Finance and Operations entity and want to see it in Dataverse, do I have to select Refresh entity list in Finance and Operations? Do I have to do anything in Dataverse?**

In theory, no, you don’t have to refresh the entity list. At most, you might have to either reset Internet Information Services (IIS) or restart IIS Express, depending on where Application Object Server (AOS) is running. The fact that the list of entities is accurate is cached in SysGlobalObjectCache, which is a per-process cache. Any
time that this cache doesn't indicate that the list is accurate, the list is rebuilt. The rebuild process takes about five seconds. Therefore, when you restart your AOS process (w3wp.exe or iisexpress.exe), the list will be accurate the next time that you query it from Dataverse. Additionally, although recompilation should flush the SysGlobalObjectCache cache, it might not. In that case, an AOS restart will flush it.

**Is there guidance on when to use a virtual entity and when to use dual-write?**

Guidance on when to use a virtual entity and when to use dual-write is covered in Integration between Finance and Operations apps and third-party services.

**When adding records using virtual entities is there any way to use number sequences?**

Yes, if the Finance and Operations entity can auto generate number sequences, then it will work the same way from the virtual entity.

**Why does ‘search view’ not work in Power Apps?**

If there are no fields added in the quick find view for the entity, then the search box does nothing. The workaround is to add one or more fields of the entity to the quick find view.

**The virtual entity performance is slow when a virtual entity has relationships to other entities. Is there guidance on how to avoid these issues?**

There could be several reasons why performance is slow when a virtual entity has relationships to other entities. This section will be updated as new patterns are identified. The following is currently known patterns, which can be used as a guidance.

When virtual entities have relationships to other entities, the virtual entity framework needs to query the related entities if the field select list includes the foreign key values for the related entities. By default, queries against the entities return all fields unless the caller requests a specific set of fields. The best practice is to specify a narrow select list. This can help to prevent slow performance.

An example of this issue is explained in Optimize Dataverse virtual table queries.
Business events provide a mechanism that lets external systems receive notifications from Finance and Operations applications. In this way, the systems can perform business actions in response to the business events.

Business events occur when a business process is run. During a business process, users who participate in it perform business actions to complete the tasks that make up the business process.

A business action that a user performs can be either a workflow action or a non-workflow action. Approval of a purchase requisition is an example of a workflow action, whereas confirmation of a purchase order is an example of a non-workflow action. Both types of actions can generate business events that external systems can use in integration and notification scenarios.

**Prerequisites**

Business events can be consumed using Microsoft Power Automate and Azure messaging services. Therefore, customers must bring their subscriptions to such assets to use business events.

**IMPORTANT**

Business events must not be considered a mechanism for exporting data. By definition, business events are supposed to be lightweight and nimble. They aren't intended to carry large payloads to fulfill data export scenarios.

**Business events that are implemented**

Business events are implemented in some business processes out of the box. These business events include both workflow and non-workflow business events. For more information, see Application business events, Workflow business events, and Alerts as business events.

A developer must use extensions to implement new business events. For more information, see Business events developer documentation.

**Business event catalog**

The business events catalog can be accessed from System administration > Set up > Business events. The business event catalog lists the business events that are available in the instance that you're using. The catalog is useful because it shows which business events are available, and you can filter it by category, business event ID, and name.

The category of a business event identifies its source. Business events that originate from the workflow system are assigned to the Workflow category. For business events that originate from other modules, the module name is used as the category name.

The business event catalog is built during database synchronization at the time of deployment. Therefore, users should see the complete list of business events in the catalog. However, if an explicit update of the catalog is required, you can select Manage > Rebuild business events catalog.
For each business event, the business event catalog shows a description. This description can help you better understand the business event and its context in the business process. The catalog also shows the list of data fields that will be sent out in the event.

In scenarios where external integration systems require the schema of the payload for a business event during development, you can select **Download schema** to download the JavaScript Object Notation (JSON) schema.

In summary, the business event catalog helps identify the business events that are required for an implementation. It also helps identify the schema for each business event.

The next step is to manage the endpoints.

**Business events parameters and processing**

The application allocates dedicated batch threads to process business events in near real time. The maximum number of threads cannot exceed the total threads available in the system (**System administration > Server configuration**). Because threads are a shared resource for all batch processing, care must be taken when deciding to change the thread allocation for business events. The total threads allocated for business events is controlled using a parameter in the business events parameter table. This setting is not exposed from the user interface (UI), so a support case must be created to get this count changed in production environments as this will need database access.

---

**IMPORTANT**

There may be reliability issues with dedicated batch threads. Microsoft is working to resolve this issue and as a result, we recommend that you schedule the manual batch job to process business events, as explained below. We will update this topic when this issue is resolved.

---

The business events batch processing job is available as a workaround to mitigate issues with the dedicated processing, if needed. The batch job can be enabled and scheduled from the **Business events parameters** page.

In the event of an error while sending business events to its end point, the system retries to send the business events three times with an interval of one second per retry. This is the default setting that can be changed on the **Business events parameters** page.

The number of endpoints that can subscribe to the same business event in a legal entity is limited to ten by default. This can be changed on the **Business events parameters** page.
The out-of-the-box default settings for the above described parameters can be restored on the Business events parameters page.

Activating business events

Business events in the business event catalog aren’t active by default. From the catalog, you can activate any business events that you require. Select one or more business events, and then select **Activate**.

Business events can be activated either in all legal entities or in specific legal entities. If you leave the Legal entity field blank, the selected business events will be activated in all legal entities. If a business event is required only for specific legal entities, it must be configured separately for each legal entity.

Endpoints must be assigned to the business events that are activated. See **Manage business event endpoints** for additional information on setting up and managing endpoints.

When business events occur as business processes are run, the system will do outbound processing only for business events that have been activated.

After business events are activated, they appear on the **Active events** tab.

From the **Active events** tab, you can inactivate business events. The system won’t do outbound processing for inactivated events.
After business events are inactivated, they appear on the **Inactive events** tab.

Business events can be inactivated when processing of business events must be paused for a period because of specific system maintenance activities in the integration landscape.

When business requirements change, some business events might no longer be required. In this case, you can inactivate them instead of deleting them from the list of active events. This approach is useful if the history of errors for the business events must be preserved. Inactivated business events can be deleted later, when there is no longer a business need to keep them inactivated.

**Errors**

While the system does outbound processing of business events, errors can occur. These errors might prevent the system from successfully delivering a business event to the endpoint. If an error occurs, the system retries several times to successfully process the business event. However, if all attempts are unsuccessful, the business event is saved in an error log.

Error logs can be accessed from the **Active events**, **Inactive events**, and **Errors** tabs. The **Errors** tab shows all errors across all business events, whereas the other two tabs show errors in the context of a specific business event.

You can do on-demand outbound processing on each error by using the **Resend** action. This action invokes the outbound processing logic. This logic includes retries. If the outbound processing is still unsuccessful, the error is logged in the error log. In this case, the **Last process time** field on the **Errors** tab indicates when the last attempt to process the event occurred.

If an error can’t be successfully processed, you can use the **Download payload** option to download the payload from the event for offline processing, as you require.

**NOTE**

If an endpoint is deleted and a new endpoint is associated with business events, all errors that are associated with the business events can still be resent. In this case, the system will do outbound processing to send to the new endpoint that is associated with the corresponding business event. This functionality allows for graceful recovery from misconfiguration or other error states.

**Business event consumption models**

The integration requirements and integration solution design for implementations vary. The integration requirements play a role in identifying the consumption model for business events. In summary, you must consider the following points when you design integrations that use business events:
Business events can be consumed using Power Automate, Service Bus, Event Grid, or other endpoint types.

Customers must bring their own subscriptions to use Power Automate, Service Bus, Event Grid, or other endpoint types.

A business event can be activated in all legal entities or in specific legal entities.

A business event can be sent to a unique endpoint or multiple endpoints.

Power Automate can directly subscribe to business events.

### Idempotency

Business events enable idempotent behavior on the consuming side by having a control number in the payload. The control number is an upwardly increasing number, which can be tracked by the consuming application to detect duplication and/or out of order delivery. The control number cannot be misread as the sequence number because the control number cannot be sequential. There can be gaps in the numbering space.

### Filtering in Azure Event Grid and Azure Service Bus

Azure Service Bus and Azure Event Grid supports subscribing to topics by specifying criteria on the incoming message. For more information, see [Topic filters and actions](#) and [Understand event filtering for Event Grid subscriptions](#).

A business event that is sent to an Azure Service Bus or Azure Event Grid has the following fields made available for this purpose. Subscribers can use this information to subscribe to more specific topics as required.

- **Category** – This is the business event category as displayed in the business event catalog. This is useful as a filter criterion when a common topic is used for receiving business events from multiple categories and subscribers want to only receive business events for the category that they are interested in.

- **Business event ID** – This is the class name of the business event implementation as displayed in the business event catalog. This uniquely identifies the business event (not the instance of the business event) and thus helps in validation of received business events on the consumer side to ensure the expected business event is what is being received and processed.

- **Legal entity** – This is the legal entity in which the business event happened. This is a useful information to base the consuming logic on if the processing and distribution of business events on the consumption side must be driven by a legal entity.

<table>
<thead>
<tr>
<th>REQUIREMENT</th>
<th>PRIVILEGE</th>
<th>DUTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only certain users must have access to view the business events catalog.</td>
<td><strong>BusinessEventsCatalogView</strong></td>
<td>None</td>
</tr>
<tr>
<td>Only certain users must have access to activate business events.</td>
<td><strong>BusinessEventsCatalogMaintain</strong></td>
<td>None</td>
</tr>
<tr>
<td>REQUIREMENT</td>
<td>PRIVILEGE</td>
<td>DUTY</td>
</tr>
<tr>
<td>---------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>Only certain users must have access to create and manage endpoints.</td>
<td>Business events security privilege</td>
<td>Business events security duty</td>
</tr>
<tr>
<td>Users must only be able to subscribe to business events which they have been granted access to from external applications like Power Automate.</td>
<td>Subscribe to business events from service</td>
<td>None</td>
</tr>
<tr>
<td>Only certain users must be able to view the business events security setup.</td>
<td>BusinessEventsCatalogSecuritySetUpView</td>
<td>None</td>
</tr>
<tr>
<td>Only certain users must be able to manage business events security.</td>
<td>Maintain business events catalog security</td>
<td>None</td>
</tr>
</tbody>
</table>

These privileges can be added to the required duties to grant corresponding roles appropriate access levels.

**Enabling role-based security for business events**

Role-based security for business events must be enabled via Feature management.

1. Go to System administration > Feature management.
2. Select the Business events catalog security feature.
3. Enable the feature.
4. Go to the business events catalog via System administration > Set up > Business events > Business events catalog.
5. The Security tab in the catalog is where a business event must be mapped to one or more roles. You must complete the configuration as required.
6. Enable security by selecting the Enable menu button on the Security menu on the top navigation pane. An informational message will confirm if security is enabled or disabled.
7. Modify the necessary security role to add the appropriate privilege or the duty based on security noted in the informational message.

**Subscribe to business events from service**

Users having access to the privilege Subscribe to business events from service via their roles will be able to only see and subscribe to business events that have been assigned to their roles, which is described below. The organizational assignments that are done, if any, as part of role-based security is honored in the context of business events by letting users to only subscribe to business events in the organizations to which they have access to via their roles. This behavior is effective using any service calls like from Power Automate or Logic Apps.

**Backward compatibility**

To ensure backward compatibility, the following behavior must be understood.

- Role-based security for business events will be disabled by default.
- Even if the feature is enabled in Feature management, role-based security will not take effect.
- Role-based security must be explicitly enabled in the business events catalog via the Security menu.
- After role-based security is enabled completely, security will be enforced henceforth. This will mean that any user with administration role will not notice any change in behavior. However, any non-admin users...
will either only see business events to which their roles were assigned to in the business events catalog security configuration or they will not see any business events because their roles were not assigned to any business events.

NOTE

To ensure uninterrupted functionality, it is important to understand the backward-compatibility behavior described above before you enable security on business events.
Endpoints let you manage the destinations that business events are sent to. Business events in Finance and Operations apps support the following endpoint types.

<table>
<thead>
<tr>
<th>ENDPOINT TYPE</th>
<th>TUTORIAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Azure Service Bus Queue</td>
<td>Business events and Azure Service Bus</td>
</tr>
<tr>
<td>Azure Service Bus Topic</td>
<td>Business events and Azure Service Bus</td>
</tr>
<tr>
<td>Azure Event Grid</td>
<td>Business events and Azure Event Grid</td>
</tr>
<tr>
<td>Azure Event Hub</td>
<td>Business events and Azure Event Hubs</td>
</tr>
<tr>
<td>Azure Blob Storage</td>
<td>Not applicable</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Microsoft Power Automate</td>
<td>Business events and Microsoft Power Automate</td>
</tr>
<tr>
<td>Dataverse</td>
<td>Subscribe to events in Dataverse</td>
</tr>
</tbody>
</table>

Endpoints can be created for these messaging and event brokers out of the box. Some scenarios might require multiple endpoints for organized distribution of business events to consumers. You can create multiple endpoints to support these scenarios.

The Microsoft Azure–based endpoints must be in the customer's Azure subscription. For example, if Event Grid is used as an endpoint, the endpoint must be in the customer's Azure subscription.

A Finance and Operations app doesn't provision the endpoints. The endpoints must be created separately and provided to the app. The app then sends events to the endpoints that are provided. Customers might incur additional costs if they use these endpoints in their Azure subscription.

### Subscribing to Finance and Operations apps events from Dataverse

**IMPORTANT**

Before you subscribe to Finance and Operations apps business events and data events in Microsoft Dataverse, you must enable the Microsoft Power Platform integration. For information about how to enable the Microsoft Power Platform integration for a Finance and Operations apps environment, see [Enable the Microsoft Power Platform integration](#).

After the Microsoft Power Platform integration is enabled, you can subscribe to Finance and Operations apps business events and data events from Dataverse. Subscription enables the following capabilities:

- Consistent behavior across events from multiple applications in Dataverse
- Application Lifecycle Management (ALM) for the Dataverse solution to consistently consume events from Finance and Operations apps
- Registration of plug-ins and software development kit (SDK) steps on Finance and Operations apps events in
When the Microsoft Power Platform integration is enabled for a Finance and Operations apps environment, endpoints that are created for business events are synced with the linked Microsoft Power Platform environment for endpoint types that are supported in Dataverse. The endpoints can then be used in Microsoft Power Platform. When the endpoints are synced, business events that are sent from Finance and Operations apps are proxied through Dataverse to the endpoint.

The following table shows the mapping between the Finance and Operations apps and Dataverse implementations of the endpoints.

<table>
<thead>
<tr>
<th>FINANCE AND OPERATIONS APPS ENDPOINT TYPE</th>
<th>DATaverse SERVICE ENDPOINT TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Azure Service Bus Queue</td>
<td>Azure Service Bus of type Queue</td>
</tr>
<tr>
<td>Azure Service Bus Topic</td>
<td>Azure Service Bus of type Topic</td>
</tr>
<tr>
<td>Azure Event Grid</td>
<td>Azure Event Grid</td>
</tr>
<tr>
<td>Azure Event Hub</td>
<td>Azure Event Hub</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Webhook</td>
</tr>
<tr>
<td>Azure Blob Storage</td>
<td>Not supported in Dataverse</td>
</tr>
<tr>
<td>Microsoft Power Automate</td>
<td>Asynchronous callback registration</td>
</tr>
<tr>
<td>Dataverse</td>
<td>Plug-in or SDK step registration</td>
</tr>
</tbody>
</table>

If an endpoint type isn't supported in Dataverse, or if the Microsoft Power Platform integration isn't enabled, the endpoint will continue to send the event from Finance and Operations apps instead of sending it through Dataverse.

**Viewing or creating mapped endpoints in Dataverse**

When a new endpoint is added in Finance and Operations apps, it's synced to Dataverse. It's then available for use in Dataverse in the `ServiceEndpoint` table. You can also create the endpoint directly in Dataverse in the `ServiceEndpoint` table. If the service endpoint is created by subscribing to a Finance and Operations apps event, it will automatically be made available to Finance and Operations apps and can be viewed on the `Endpoints` tab of the `Business events` page. This behavior is applicable to the following mapped endpoint types:

- Azure Service Bus Queue
- Azure Service Bus Topic
- Azure Event Grid
- Azure Event Hub

For more information about the `ServiceEndpoint` table, see [ServiceEndpoint table/entity reference](#).

**Microsoft Power Automate endpoints**

The Microsoft Power Automate endpoint type isn't made available for setup directly in Finance and Operations apps. This endpoint type is used for subscriptions that are created and sent directly from a flow in Power Automate.
The endpoint is created on the **Endpoints** tab of the **Business events** page in Finance and Operations apps when you subscribe to a Finance and Operations apps business event or data event in Power Automate. For more information about how to subscribe to business events and data events in Power Automate, see **Business events in Microsoft Power Automate**.

**Microsoft Dataverse endpoints**

The **Dataverse** endpoint type also isn’t available for manual setup in Finance and Operations apps. The endpoint is created when a plug-in or an SDK step is registered on a Finance and Operations apps business event or data event in Dataverse. When the step is registered, it becomes visible as an endpoint in the list on the **Endpoints** tab of the **Business events** page in Finance and Operations apps.

The business event registration itself will also be listed on either the **Business event catalog** tab or the **Data event catalog** tab of the **Business events** page in Finance and Operations apps, depending on registration. In this way, Finance and Operations apps users can learn which business event or data event has a plug-in or SDK step registered in Dataverse. They can also learn the reason why the event is active in Finance and Operations apps.

Finance and Operations apps events can be subscribed to directly in Dataverse by using the tools in the Dataverse toolset, such as the Power Platform Tools extension for Visual Studio. For more information about this extension, see **Install Power Platform Tools**. These subscriptions will appear on the **Business event catalog** tab of the **Business events** page in Finance and Operations apps.

Some attributes of service endpoints in Dataverse, such as the name and description, can be updated. These updates will also be reflected in Finance and Operations apps. However, updates that change the service endpoint type will be prevented if the service endpoint is used with Finance and Operations apps events. These updates include a change from a Service Bus topic to a Service Bus queue, which Dataverse usually allows. This behavior helps ensure design simplicity and consistency, because Finance and Operations apps don’t allow these updates to endpoints after they have been created.

After service endpoints are created, Dataverse doesn’t allow them to be deleted if they are being used. This limitation also applies to service endpoints that are used by Finance and Operations apps events. Any attempt to delete one of these endpoints will cause an error, and deletion will be prevented.

For more information about how to subscribe to Finance and Operations apps business events in Dataverse, see **Subscribe to events in Dataverse**.
This topic lists application business events.

### Procure to pay

<table>
<thead>
<tr>
<th>BUSINESS EVENT</th>
<th>DESCRIPTION</th>
<th>MODULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor invoice matched</td>
<td>This event is triggered when invoice matching validation is completed for a vendor invoice as part of the Procure to pay process.</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Vendor invoice posted</td>
<td>This event business event is triggered when a user posts a vendor invoice as part of the Procure to Pay process.</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Vendor payment posted</td>
<td>This event is triggered when a user posts a vendor payment as part of the Procure to pay process.</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Invoice register journal posted</td>
<td>This event is triggered when a user posts an invoice register journal as part of the Procure to pay process.</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Invoice journal posted</td>
<td>This event is triggered when a user posts an invoice journal as part of the Procure to pay process.</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Invoice approval journal posted</td>
<td>This event is triggered when a user posts an invoice approval journal as part of the Procure to pay process.</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Purchase order confirmed</td>
<td>This event is triggered when a purchase order is confirmed by a vendor. One of the following actions triggers the event: the user manually confirms a purchase order in the user interface for purchase orders, when the purchase order confirmation is executed in a batch, or when the confirmation is executed programmatically in intercompany scenarios. In scenarios where vendor collaboration is used, and the vendor collaboration policy is set to autoconfirm a purchase order, the trigger occurs when the <strong>Accept</strong> button is clicked on the <strong>Purchase order confirmation</strong> page in the <strong>Vendor collaboration</strong> portal.</td>
<td>Procurement and sourcing</td>
</tr>
</tbody>
</table>
### Purchase order received

This event is triggered when goods or services are registered as received against one or more purchase orders. One of the following actions triggers the event: a product receipt is generated for one or more purchase orders manually in the user interface for purchase orders and product receipts, when product receipts are generated in a batch, or when product receipts are generated programmatically in intercompany scenarios.

### Quote to cash

<table>
<thead>
<tr>
<th>BUSINESS EVENT</th>
<th>DESCRIPTION</th>
<th>MODULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invoice is created from a sales order</td>
<td>This event is triggered when a user posts a sales order invoice as part of the Quote to Cash process.</td>
<td>Accounts receivable</td>
</tr>
<tr>
<td>Free text invoice posted</td>
<td>This event is triggered when a user posts a free text invoice as part of the Quote to Cash process.</td>
<td>Accounts receivable</td>
</tr>
<tr>
<td>Payment posted</td>
<td>This event is triggered when a user posts a payment as part of the Quote to Cash process.</td>
<td>Accounts receivable</td>
</tr>
<tr>
<td>Transaction is written off</td>
<td>This event is triggered when a user writes off a customer transaction as part of the Quote to Cash process.</td>
<td>Accounts receivable</td>
</tr>
<tr>
<td>Collection status of a transaction changed</td>
<td>This event is triggered when a user updates the collection status of a transaction as part of the Quote to Cash process.</td>
<td>Accounts receivable</td>
</tr>
<tr>
<td>Interest note posted</td>
<td>This event is triggered when a user posts an interest note as part of the Quote to Cash process.</td>
<td>Accounts receivable</td>
</tr>
<tr>
<td>Collection letter created</td>
<td>This event is triggered when a user creates a collection letter for a customer as part of the Quote to Cash process.</td>
<td>Credit and collections</td>
</tr>
</tbody>
</table>
Workflow business events are generated at various points in the processing of a workflow.

**Workflow construction**

To construct a workflow, a developer can define workflows components in metadata and code in the Visual Studio tools.

An administrator can create workflows in the web client and then design them in the workflow designer. For more information, see [Create workflows](#).

**Workflow components**

Workflow components are defined in metadata as:

- **Workflow types** - Also known as templates, workflow types define the elements allowed in a workflow. The administrator decides which elements are actually used when they create the workflow design.
  - In the Application Explorer, go to AOT > Business Process and Workflow > Workflow Types.
- **Workflow elements** - Workflow elements are the executable pieces that make up a workflow. For details, see [Workflow elements](#).
  - Tasks (aka manual tasks) - Go to AOT > Business Process and Workflow > Workflow Tasks.
  - Approvals - Go to AOT > Business Process and Workflow > Workflow Approvals.
  - Automated tasks - Go to AOT > Business Process and Workflow > Workflow Automated Tasks.

**Workflow runtime**

After a workflow is submitted by a user, it is added to a queue and run using the **Workflow message processing** batch job. As the workflow runs, it will progress through all the **connected workflow elements** until it reaches the end. When the workflow runtime encounters a **manual task element**, it will create a work item for the **user assigned to the task**. When the workflow runtime encounters an **approval element**, it will create a work item for each **user assigned to each approval step**.

**Workflow business event categories**

There are five different categories of workflow business events. The category will show up in Microsoft Power Automate to help with event selection.
- **Category: Workflow type**
  - These events will fire on workflow events like started and completed. All workflow instances will be represented in this category.
  - **ID format** - "Workflow_" + Workflow name + Workflow instance ID, for example "Workflow_BudgetPlanReview_000002"
  - **Name format** - Workflow label + "(" + Workflow instance ID ")", for example "Prepare department budget (000002)"

- **Category: Workflow element started**
  - These events will fire when a workflow element is started. All enabled workflow elements within a workflow instance will be represented in this category.
  - **ID format** - "Workflow_" + Workflow name + Workflow instance ID + "." + Workflow element name + ".Started", for example "Workflow_BudgetPlanReview_000002_BudgetActivateBudgetPlanChild_Started"
  - **Name format** - Workflow label + "(" + Workflow instance ID ") - " + Workflow element label, for example "Prepare department budget (000002) - Activate associated budget plan"

- **Category: Workflow element**
  - These events will fire on workflow element events other than started, such as completed. All enabled workflow elements within a workflow instance will be represented in this category.
  - **ID format** - "Workflow_" + Workflow name + Workflow instance ID + "." + Workflow element name, for example "Workflow_BudgetPlanReview_000002_BudgetActivateBudgetPlanChild"
  - **Name format** - Workflow label + "(" + Workflow instance ID ") - " + Workflow element label, for example "Prepare department budget (000002) - Activate associated budget plan"

- **Category: Workflow external task**
  - These events will fire when a workflow automated task element is started. All enabled workflow automated task elements within a workflow instance will be represented in this category.
  - **ID format** - "Workflow_" + Workflow name + Workflow instance ID + "." + Workflow element name + ".ExternalTask", for example "Workflow_BudgetPlanReview_000002_BudgetActivateBudgetPlanChild_ExternalTask"
  - **Name format** - Workflow label + "(" + Workflow instance ID ") - " + Workflow element label, for example "Prepare department budget (000002) - Activate associated budget plan"
• **Category: Workflow workitem**
  - These events will fire when a workflow work item is created for a user. All enabled workflow tasks and workflow approvals within a workflow instance will be represented in this category.
  - **ID format** - "Workflow._" + Workflow name + Workflow instance ID + "," + Workflow element name + "," + WorkItem", for example
    "Workflow_BudgetPlanReview_000002_BudgetActivateBudgetPlanChild_WorkItem"
  - **Name format** - Workflow label + " (" + Workflow instance ID ") - " + Workflow element label, for example "Prepare department budget (000002) - Activate associated budget plan"

### Completion of a work item in Power Automate

Workflow business events are a good target for triggering approval flows. The **workflow workitem** event can be used in conjunction with the validate and complete OData actions to facilitate completion of a work item in Power Automate.

An approval or task work item can be completed in Power Automate using the following steps:

1. **Trigger the Power Automate using the when a business event occurs trigger** targeting the appropriate **workflow workitem** event.
2. **Validate that the workflow workitem contains a valid set of information so it is ready for completion by calling the Validate method on the WorkflowWorkItems entity.**
3. **If the workitem is not ready for completion, then send a notification to the assigned user to let them know that there is a workitem that needs their attention.**
4. **If the workitem is ready for completion, then request a response from the assigned user by sending the available response options to the user.**
5. **After a response is provided, complete the workitem with that response by calling the Complete method on the WorkflowWorkItems entity.**

To enable external completion of work items, the work item action manager class needs to implement the IValidateWorkflowWorkItemAction interface. The standard WorkflowWorkItemActionManager class has implemented this interface. In Platform update 32, the TrvWorkflowWorkItemActionManager class was updated to implement the IValidateWorkflowWorkItemAction interface. Use the existing IValidateWorkflowWorkItemAction implementations as examples to notify updates about other WorkflowWorkItemActionManager classes.

For a step-by-step guide to setting up work item completion in Microsoft Power Automate, see **Consume workflow approval business events**.

### Templates for work item completion in Power Automate

The following templates for work item completion in Power Automate are available:

- **Complete Dynamics 365 for Finance and Operations workflow work items (PU26)**
- **Complete Dynamics 365 for Finance and Operations workflow work items (PU29)**

The Platform update 29 version gets completion options from the business event payload. These options were added in Platform update 29, and presented to the user via the approval action.

### Troubleshooting workflow business events

#### Troubleshooting workflow issues

Ensure that the workflow is running correctly and creating work items as expected. If the workflow doesn’t work inside the application so that state changes are occurring, then the events won’t occur. Adjust the workflow configuration as needed. If needed, review the workflow details in the **Workflow History form**.
Troubleshooting Power Automate issues

Ensure that the Power Automate subscription is available in the System administration > Setup > Business events > Business events catalog on the Active events tab. If the Power Automate subscription isn’t there, then check Power Automate and recreate it if needed.

Troubleshooting business events issues

Ensure that other business events are occurring by creating a Power Automate to trigger off another business event. For example, the Free Text Invoice Posted event can be triggered by simply creating a Free Text Invoice with a single line and posting it. For more information, see Troubleshoot business events.

Troubleshooting work item approval via Power Automate

If a flow is trying to handle approval for work items, but it isn’t firing, then verify these steps:

- Are the work items being created so the applicable user can see them waiting for approval in the web client?
- Is the event subscription from Flow visible in the Business Events form?
- Are the workflow configuration and the event subscription from Flow for the correct legal entity (company)?
The batch framework emits the following system business events.

<table>
<thead>
<tr>
<th>BUSINESS EVENT</th>
<th>DESCRIPTION</th>
<th>MODULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch job started</td>
<td>This event is fired when a batch job is marked as running.</td>
<td>Batch</td>
</tr>
<tr>
<td>Batch job finished</td>
<td>This event is fired when a batch job is completed.</td>
<td>Batch</td>
</tr>
<tr>
<td>Batch job failed</td>
<td>This event is fired when a batch job fails.</td>
<td>Batch</td>
</tr>
<tr>
<td>Batch job cancelled</td>
<td>This event is fired when a batch job is canceled.</td>
<td>Batch</td>
</tr>
</tbody>
</table>

**Usage**

**Batch job started and batch job finished**

The **batch job started** and **batch job finished** events can be used to monitor and identify long-running batch jobs. They can also be used to notify stakeholders if a job takes longer than expected. By default, these events are turned off in the application.

**Batch job failed**

The **batch job failed** event can be used to monitor specific batch jobs for failure, and to notify stakeholders in real time. By default, this event is turned on in the application.

**Configure batch business events**

1. Go to System administration > Inquiries > Batch jobs.
2. On the Business events tab, update the settings to raise batch business events.

The events have the following payload.

<table>
<thead>
<tr>
<th>FIELD NAME</th>
<th>FIELD LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobId</td>
<td>Batch job ID</td>
</tr>
<tr>
<td>JobDescription</td>
<td>Batch job description</td>
</tr>
<tr>
<td>JobStatus</td>
<td>Batch job status</td>
</tr>
<tr>
<td>JobOwnerEmailId</td>
<td>Batch job owner email ID</td>
</tr>
<tr>
<td>JobExecutedByEmailId</td>
<td>Batch job executed by email ID</td>
</tr>
<tr>
<td>FIELD NAME</td>
<td>FIELD LABEL</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>AdminEmailId</td>
<td>Admin user email ID</td>
</tr>
<tr>
<td>JobEndUtcDateTime</td>
<td>Job end UTC date time</td>
</tr>
<tr>
<td>BusinessEventId</td>
<td>Business event ID</td>
</tr>
<tr>
<td>ControlNumber</td>
<td>Business event control number</td>
</tr>
<tr>
<td>EventId</td>
<td>Business event instance ID</td>
</tr>
<tr>
<td>EventTime</td>
<td>Business event instance ID</td>
</tr>
<tr>
<td>MajorVersion</td>
<td>Major version</td>
</tr>
<tr>
<td>MinorVersion</td>
<td>Minor version</td>
</tr>
</tbody>
</table>

To get the current catalog and schema of business events, go to **System administration > Setup > Business events**.
There are two kinds of alerts that can be configured by users. These are change-based alerts and due date alerts. For more information about the alerts functionality, see Alerts.

The change-based alerts and due date alerts can be configured to send out a business event as a mechanism to notify or trigger external applications or systems. This allows alerts to participate in advanced user notification scenarios and also in business process integration across systems.

To generate a business event from an alert, in the Create alert rule dialog box, set Alert me with > Send externally to Yes.

In order for alerts to be processed, the batch processes for change-based and/or due-date alerts should be set for batch processing for due-date events. For more information, see Batch processing for due-date events.

The business event for the change-based alert and/or the due date alert must also be active for the alert to be sent out as a business event. To learn more about the activation process, see Activating business events.
Data events are events that are based on changes to data in Finance and Operations apps. Create, update, and delete (CUD) events can be enabled for each entity. For example, if the Create event is enabled for the Purchase order headers V2 entity, an event notification is emitted every time that a new purchase order is created in the database.

All standard and custom entities in Finance and Operations apps that are enabled for Open Data Protocol (OData) can emit data events. In the data event catalog, each event for an entity is listed as a data event that subscriptions can be established for. The concept of activating the data event and associating it with an endpoint resembles the concept of business events. When a data event occurs, the payload of the event contains the corresponding entity record.

Data events are available only in environments that the Microsoft Power Platform integration is enabled for. For more information, see Enable the Microsoft Power Platform integration.

Data event catalog

To access the data event catalog, select the Data event catalog tab on the Business events page (System administration > Setup > Business event). The data event catalog provides a complete list of the available data events in the Finance and Operations apps environment. For each data event, the list shows the category, event ID, and name. It also indicates whether the event is company-specific. You can filter the list by category and data event ID.

Activating data events

By default, data events are inactive. To activate a data event from the data event catalog, select it in the list, and then select Activate.
Data events can be activated either in all legal entities or in specific legal entities. In the **Configure new data event** dialog box, in the **Legal entity** field, select the legal entity that you want to activate the data events in. If you leave the **Legal entity** field blank, the selected data events will be activated in all legal entities. If a data event is required in multiple specific legal entities, it must be configured separately for each legal entity.

Only company-specific data events can be configured for specific legal entities. When you configure data events that aren’t company-specific, the **Legal entity** field isn’t editable, and the data events are enabled for all legal entities.

To activate a data event, select a configured endpoint. For information about how to configure endpoints business events and data events, see **Manage business event endpoints**.

When you activate a data event, it’s added to the list on the **Active data events** tab of the **Business events** page and becomes available for subscription through the selected endpoint.

### Deactivating data events

You can deactivate data events from the **Active data events** tab of the **Business events** page. To deactivate a data event, select it in the list, and then select **Deactivate**. The data event is removed from the list on the **Active data events** tab and added to the list on the **Inactive data events** tab.

Like business events, data events can be deactivated when business event processing is temporarily paused. A temporary pause might be required because of system maintenance, bulk data imports, or bulk data processing. Bulk data processing where data events are enabled on related data entities can send a high volume of data events that might not be required. This situation can affect system performance.

When data events are no longer required to meet business requirements, you can delete them from the list on the **Active data events** or **Inactive data events** tab. In this case, the data events are removed from the list, and all error history for them is deleted. If the history of errors for a data event must be preserved, you can deactivate the data event instead of deleting it. For more information about error logs for business events and data events, see **Errors**.

### Download the data event schema

The **Data event catalog** tab on the **Business events** page also shows the fields that are passed to a data event and that make up the event schema. The information that is shown includes the field name and label. You can download the JavaScript Object Notation (JSON) schema for an event by selecting **Download schema**.

<table>
<thead>
<tr>
<th>Data event catalog tab</th>
<th>Event schema fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business event catalog</td>
<td>Data event schema</td>
</tr>
<tr>
<td>Endpoint type</td>
<td></td>
</tr>
<tr>
<td>Endpoint type</td>
<td></td>
</tr>
<tr>
<td>Endpoint type</td>
<td></td>
</tr>
</tbody>
</table>

**End of document**
This capability is helpful when external integration systems require the schema of the payload for a business event during development.
The Finance and Operations connector and Microsoft Dataverse connector are available for consuming business events in Microsoft Power Automate. The Finance and Operations connector has a **When a Business Event occurs** trigger. The Dataverse connector has a **When an action is performed** trigger. Either of these triggers can be used to subscribe to any of the business events that are available in Finance and Operations apps. Both triggers provide the same functionality, but the execution is slightly different.

The Dataverse connector lets you use the **When a row is added, modified or deleted** trigger to subscribe to data events in Finance and Operations apps. This trigger enables a Power Automate flow to be triggered by any create, update, or delete (CUD) event against a selected Finance and Operations apps entity.

**Prerequisite**

It's important that you understand business events. For more information, see the Business events documentation.

**Subscribing to business events**

**Using the Finance and Operations connector**

The Finance and Operations connector communicates directly with Finance and Operations apps to establish subscriptions, but it's triggered by Dataverse at runtime. The connector can connect to any instance of Finance and Operations apps on the Azure Active Directory (Azure AD) tenant.

After the **When a Business Event occurs** trigger is added to a flow, the following information must be provided:

- **Instance** – Specify the host name of the instance where business events occur. Environment instances should be available in the provided drop-down menu, but if an environment is not listed it can be entered as a custom value.
- **Category** – Select the category of business events. The list of unique business event categories in the business event catalog in Finance and Operations apps is shown.
- **Business event** – Select the business event that the flow should be triggered from. All the business events that are shown in the list are business events in the selected category in the Finance and Operations apps business event catalog.
- **Legal entity** – Specify the legal entity where the business event is being subscribed to. The flow will be triggered when the business event occurs in that legal entity. By default, this field is blank and the business event is subscribed to in **all** legal entities.

When the flow is saved, a subscription to the selected business event is added into the environment instance. As
Using the Dataverse connector

Business events for Finance and Operations apps are also exposed through the **When an action is performed** trigger of the Dataverse connector. This trigger exposes actions and table operations that are configured in Dataverse by using the **Catalog** and **CatalogAssignment** tables. This configuration provides a more generic business event framework in Dataverse that isn’t limited to Finance and Operations apps business events. Business events in the Finance and Operations apps business event catalog are synchronized with the Dataverse business events catalog. Therefore, you can subscribe to Finance and Operations apps business events to initiate business logic in a Power Automate flow. For more information about the catalog in the Dataverse business events framework, see **Catalog and CatalogAssignment tables**.

To use the Finance and Operations apps business events in the **When an action is performed** trigger of the Dataverse connector, the Microsoft Power Platform integration must be enabled for the Finance and Operations apps environment, so that the Finance and Operations apps environment is connected to the Dataverse environment. For more information about how to enable the Microsoft Power Platform integration for Finance and Operations apps environments, see **Enabling the Microsoft Power Platform integration**.

---

**NOTE**

The Microsoft Power Platform integration has a one-to-one connection between Finance and Operations apps and the Microsoft Power Platform environment. Because of this relationship, you can’t select among multiple Finance and Operations apps environments as the **When a Business Event occurs** trigger of the Finance and Operations connector does. The trigger automatically connects to the Finance and Operations apps environment that is selected for the Microsoft Power Platform integration.

---

After the **When an action is performed** trigger is added to a flow in Power Automate, the following information must be provided:

- **Catalog** – Select **Finance and Operations**. This exposes Finance and Operations business events as a Dataverse business events catalog.
- **Category** – Select the category of the desired business event. The list of unique business event categories in the business event catalog in Finance and Operations apps is shown.
- **Table name** – If the action is related to a specific table, select the related table. Typically, the value will be **(none)** for Finance and Operations apps business events.
- **Action name** – Select the action or business event that the flow should be triggered from. The drop-down list shows all synchronized business events in the selected category in the Finance and Operations apps business event catalog.

For more information about how to use the **When an action is performed** trigger in Power Automate, see **Trigger flows with actions**.
Subscribing to data events

Finance and Operations apps entities that are enabled as virtual entities in Dataverse are included in the **When a row is added, modified or deleted** trigger of the Dataverse connector. When you add the trigger to a flow in Power Automate, define the table name of the table that you want to trigger the flow for. The **Table name** list contains the list of all Finance and Operations apps entities that are exposed as virtual entities in Dataverse from the Finance and Operations apps environment that is connected to the Microsoft Power Platform environment through the Microsoft Power Platform integration. For information about how to enable virtual entities, see [Enable Dataverse virtual entities](#).

For more information about how to use the **When a row is added, modified or deleted** trigger in the Dataverse connector, including information about advanced options, see [Trigger flows when a row is added, modified, or deleted](#).

Unsubscribing from business events

If the trigger is deleted or the flow is turned off, the business event endpoint is automatically deleted.

Adjusting flow parameter limits

Multiple flows can subscribe to the same business event in different legal entities or in the same legal entity. The default endpoint limit per event is ten. You can adjust the **Endpoints allowed per event** setting on the **Business event parameters** page as you require.

Other ways to consume business events in Power Automate

The previous section explains how you can subscribe to business events directly from Power Automate by using the trigger in the connector. However, you can also consume business events in Microsoft Power Automate from Microsoft Azure Event Grid, by using the [Event Grid connector for Microsoft Power Automate](#).

Event Grid might be a viable approach for consuming business events in Power Automate if it’s already being used for other integrations in an implementation. If a business event in the same legal entity must trigger multiple flows, you should consider consuming the business event from Event Grid.
This approach is applicable to any messaging or event platform that is used as an endpoint for business events, provided that a connector is available for it in Power Automate.

For information about how to use business events in Microsoft Flow, see Consume business events in Microsoft Flow.
This topic walks you through the development process and best practices for implementing business events.

What is a business event, and what isn’t a business event?

This question comes up every time that we start to think about use cases where business events can help. Is the creation of a vendor a business event? Is confirmation of a purchase order a business event? Is it a business event if you capture the event at the table level? Or should business events be captured only at the business logic level in a business process? These questions aren’t just valid, but they are also a key topic of discussion when a solution is planned and architected for integration. The following guidelines can help with this thought process and decision making.

Intent

The intent behind capturing a business event must be clearly understood. In other words, what is the reason for capturing the business event, and how it will be used by the recipient?

If your intent is to capture a business event so that you can take a business action outside Dynamics 365 Finance and Operations apps in response to a business event that occurs in Finance and Operations, you have a good use case for business events. The business action that is taken in response to the business event can be to notify users about the business event and/or to call into another business application to take a business action, such as creation of a sales order. It's important that you look at the business action generically and not base the need for a business event on the type of business action that will be taken.

If your intent is to transfer data to a recipient and, in effect, realize a data export scenario, you don't have a good use case for business events. In fact, the use of business events for data transfer scenarios is a misuse of the business events framework. Such scenarios must continue to use data export mechanisms that are already available in data management.

Fidelity

When the intent is clear, and a legitimate need for a business event is established, the next step is to evaluate the approach that must be used to capture the business event. This section summarizes the approach that must be evaluated.

Regardless of the approach that is used, the fidelity of business events is significant, because it helps guarantee that the following aspects are taken care of. Therefore, it must influence the design choice for implementing the business event. However, the design choice that you make to implement a business event must not influence the concept of business events. In other words, the chosen design must not be used as a decision-making tool, to determine whether an event is a business event. The intent must be used to make those decisions.

- **Durable business events** – No false business events should be sent to the recipient. If a purchase order confirmation business event is sent out, the recipient expects and must trust that the purchase order was really confirmed. The design choice must help guarantee this transactional nature. Therefore, you must not make a design choice that violates the recipient's expectations.

- **Targeted** – Business events must be designed to optimize the consumption story for the recipient. In other words, you should make it as easy as possible for the recipient to consume business events. Therefore, business events must be as specific as possible and must be targeted to specific use cases. They must not be generic, so that the consumer has to determine what the business event is for by trying to understand the
payload. The design choice must allow for preservation of targeted business events.

- **Noiseless** – The design should include very little effort to filter out noise. To make business events very specific, avoid writing filtering logic to filter out conditions that don’t match the expected business event. The chosen approach must help guarantee that the business event is implemented in code at a sufficiently specific point so that no filtering of noise is required. Any attempt to filter noise by adding logic can affect performance and might also become complicated in specific use cases.

The following table compares capture at the business and table levels.

<table>
<thead>
<tr>
<th>CAPTURE AT THE BUSINESS LOGIC LEVEL</th>
<th>CAPTURE AT THE TABLE LEVEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capture at this level helps guarantee durability because it occurs in the transaction.</td>
<td>Capture at this level helps guarantee durability because it occurs in the transaction.</td>
</tr>
<tr>
<td>Capture at this level allows for targeted business events.</td>
<td>Because events are captured at a lower level, it's difficult to provide targeted business events.</td>
</tr>
<tr>
<td>It's easy to remain noiseless.</td>
<td>It's difficult to remain noiseless unless additional effort is made to implement sound logic that filters out noise.</td>
</tr>
<tr>
<td>Capture at this level provides additional context for the business process, and can significantly improve the durability and quality of the payload.</td>
<td>Because events are captured at a lower level, business process context is probably lost.</td>
</tr>
</tbody>
</table>

**NOTE**

In general, if you implement business events at the table level, you might face other challenges, in addition to the challenges that are described in the preceding table. For example, if the business logic is run via a stored procedure that updates data in the underlying table, the business event might not even be generated, because it was implemented in the table insert method in X++. You might encounter additional challenges in specific use cases. Therefore, we don't recommend that you implement business events at the table level.

### Implement a business event

The process for implementing a business event and sending it is fairly straightforward.

1. Build the contract.
2. Build the event.
3. Add code to send the event.

Two classes must be implemented:

- **Business event** – This class extends the `BusinessEventsBase` class. It supports constructing the business event, building the payload, and sending the business event.

- **Business event contract** – This class extends the `BusinessEventsContract` class. It defines the payload of the business event and allows for population of the contract at runtime.

**BusinessEventsBase extension**

**Naming convention**

The names of business events should follow the pattern `<noun or noun phrase> <past tense action>BusinessEvent`. The `<noun or noun phrase>` part of the name should comply with existing definitions for application area prefixes.

**Examples**
VendorInvoicePostedBusinessEvent
CollectionLetterSentBusinessEvent

Implementation
The process of implementing an extension of the BusinessEventsBase class is straightforward. It involves extending the BusinessEventsBase class, and implementing a static constructor method, a private new method, methods to maintain internal state, and the buildContract method.

1. Implement a static newFrom<my_buffer> method. The <my_buffer> part of the method name is typically the table buffer that is used to initialize the business event contract.

```java
static public SalesInvoicePostedBusinessEvent newFromCustInvoiceJour(CustInvoiceJour _custInvoiceJour)
{
    SalesInvoicePostedBusinessEvent businessEvent = new SalesInvoicePostedBusinessEvent();
    businessEvent.parmCustInvoiceJour(_custInvoiceJour);
    return businessEvent;
}
```

2. Extend the BusinessEventsBase class.

```java
[BusinessEvents(classStr(SalesInvoicePostedBusinessEventContract),
    'AccountsReceivable:SalesOrderInvoicePostedBusinessEventName',
    'AccountsReceivable:SalesOrderInvoicePostedBusinessEventDescription',
    ModuleAxapta::SalesOrder)]
public class SalesInvoicePostedBusinessEvent extends BusinessEventsBase
```

Note the BusinessEvents attribute. This attribute provides the business events framework with information about the business event's contract, name, and description. It also provides the module that the business event is part of. Labels must be defined for the name and description arguments. However, these labels should be referenced without the at symbol (@) to avoid storing localized data.

3. Implement a private new method. This method is called only from the static constructor method.

```java
private void new()
{
}
```

4. Implement private parm methods to maintain internal state.

```java
private CustInvoiceJour parmCustInvoiceJour(CustInvoiceJour _custInvoiceJour = custInvoiceJour)
{
    custInvoiceJour = _custInvoiceJour;
    return custInvoiceJour;
}
```

5. Implement the buildContract method. Note that you need an EventContract stub for this step.

```java
[Wrappable(true), Replaceable(true)]
public BusinessEventsContract buildContract()
{
    return SalesInvoicePostedBusinessEventContract::newFromCustInvoiceJour(custInvoiceJour);
}
```

For extensibility, the buildContract method must have the Wrappable(true) and Replaceable(true)
Here is the complete implementation of the “Sales order invoice posted” business event.

```csharp
/// <summary>
/// Sales order invoice posted business event.
/// </summary>
[BusinessEvents(classStr(SalesInvoicePostedBusinessEventContract),
'AccountsReceivable:SalesOrderInvoicePostedBusinessEventName',
'AccountsReceivable:SalesOrderInvoicePostedBusinessEventDescription',
ModuleAxapta::SalesOrder)]
public class SalesInvoicePostedBusinessEvent extends BusinessEventsBase
{
    private CustInvoiceJour custInvoiceJour;
    private CustInvoiceJour parmCustInvoiceJour(CustInvoiceJour _custInvoiceJour =
    custInvoiceJour)
    {
        custInvoiceJour = _custInvoiceJour;
        return custInvoiceJour;
    }
    /// <summary>
    /// Creates a SalesInvoicePostedBusinessEvent from a CustInvoiceJour record.
    /// <summary>
    /// param name = "_custInvoiceJour"> CustInvoiceJour record <param>
    /// <returns>A SalesInvoicePostedBusinessEvent </returns>
    static public SalesInvoicePostedBusinessEvent
    newFromCustInvoiceJour(CustInvoiceJour _custInvoiceJour)
    {
        SalesInvoicePostedBusinessEvent businessEvent = new
        SalesInvoicePostedBusinessEvent();
        businessEvent.parmCustInvoiceJour(_custInvoiceJour);
        return businessEvent;
    }
    private void new()
    {
    }
    [Wrappable(true), Replaceable(true)]
    public BusinessEventsContract buildContract()
    {
        return SalesInvoicePostedBusinessEventContract::newFromCustInvoiceJour(custInvoiceJour);
    }
}
```

**BusinessEventsContract extension**

A business event contract class extends the `BusinessEventsContract` class. It defines and populates the payload of the business event. Although there is some variation across business events, the basic structure of the business event contract is consistent.

The process of implementing a business event contract involves extending the `BusinessEventContract` class, defining internal state, implementing an initialization method, implementing a static constructor method, and implementing `parm` methods to access the contract state.

1. Extend the `BusinessEventContract` class.

   ```csharp
   [DataContract]
   public final class SalesInvoicePostedBusinessEventContract extends
   BusinessEventsContract
   {
   }
   ```

   The class must have the `DataContract` attribute.

2. Add private variables to hold the contract state.
3. Implement a private initialization method.

```java
private void initialize(CustInvoiceJour _custInvoiceJour)
{
    invoiceAccount = _custInvoiceJour.InvoiceAccount;
    invoiceId = _custInvoiceJour.InvoiceId;
    salesId = _custInvoiceJour.SalesId;
    invoiceDate = _custInvoiceJour.InvoiceDate;
    invoiceDueDate = _custInvoiceJour.DueDate;
    invoiceAmount = _custInvoiceJour.InvoiceAmountMST;
    invoiceTaxAmount = _custInvoiceJour.SumTaxMST;
    legalEntity = _custInvoiceJour.DataAreaId;
}
```

The `initialize` method is responsible for setting the private state of the business event contract class, based on data that is provided through the static constructor method.

4. Implement a static constructor method.

```java
public static SalesInvoicePostedBusinessEventContract
newFromCustInvoiceJour(CustInvoiceJour _custInvoiceJour)
{
    var contract = new SalesInvoicePostedBusinessEventContract();
    contract.initialize(_custInvoiceJour);
    return contract;
}
```

The static constructor method calls a private `initialize` method to initialize the private class state.

5. Implement `parm` methods to access the contract state.

```java
[DataMember('InvoiceAccount'), BusinessEventsDataMember("@AccountsReceivable:InvoiceAccount")]
public CustInvoiceAccount parmInvoiceAccount(CustInvoiceAccount _invoiceAccount = invoiceAccount)
{
    invoiceAccount = _invoiceAccount;
    return invoiceAccount;
}
```

The `parm` methods should have the `DataMember('<name>')` and `BusinessEventsDataMember('<description>')` attributes. The name that you provide on the `DataMember` attribute (for example, 'InvoiceAccount') will be visible to data contract consumers. The description that you provide in the `BusinessEventsDataMember` attribute will be visible in the Business Events catalog user interface (UI) and used to describe this contract's data members.
/// <summary>
/// The data contract for a SalesInvoicePostedBusinessEvent
/// </summary>
[DataContract]
public final class SalesInvoicePostedBusinessEventContract extends BusinessEventsContract
{
    private CustInvoiceAccount invoiceAccount;
    private CustInvoiceId invoiceId;
    private SalesIdBase salesId;
    private TransDate invoiceDate;
    private DueDate invoiceDueDate;
    private AmountMST invoiceAmount;
    private TaxAmount invoiceTaxAmount;
    private LegalEntityDataAreaId legalEntity;

    /// <summary>
    /// Creates a SalesInvoicePostedBusinessEventContract from a CustInvoiceJour record.
    /// </summary>
    /// <param name="_custInvoiceJour"> CustInvoiceJour record</param>
    /// <returns>A SalesInvoicePostedBusinessEventContract </returns>
    public static SalesInvoicePostedBusinessEventContract newFromCustInvoiceJour(CustInvoiceJour _custInvoiceJour)
    {
        var contract = new SalesInvoicePostedBusinessEventContract();
        contract.initialize(_custInvoiceJour);
        return contract;
    }

    private void initialize(CustInvoiceJour _custInvoiceJour)
    {
        invoiceAccount = _custInvoiceJour.InvoiceAccount;
        invoiceId = _custInvoiceJour.InvoiceId;
        salesId = _custInvoiceJour.SalesId;
        invoiceDate = _custInvoiceJour.InvoiceDate;
        invoiceDueDate = _custInvoiceJour.DueDate;
        invoiceAmount = _custInvoiceJour.InvoiceAmountMST;
        invoiceTaxAmount = _custInvoiceJour.SumTaxMST;
        legalEntity = _custInvoiceJour.DataAreaId;
    }

    private void new()
    {
    }

    [DataMember('InvoiceAccount'), BusinessEventsDataMember('@AccountsReceivable:InvoiceAccount')]
    public CustInvoiceAccount parmInvoiceAccount(CustInvoiceAccount _invoiceAccount = invoiceAccount)
    {
        invoiceAccount = _invoiceAccount;
        return invoiceAccount;
    }

    [DataMember('InvoiceId'), BusinessEventsDataMember('@AccountsReceivable:BusinessEventInvoiceId')]
    public CustInvoiceId parmInvoiceId(CustInvoiceId _invoiceId = invoiceId)
    {
        invoiceId = _invoiceId;
        return invoiceId;
    }

    [DataMember('SalesId'), BusinessEventsDataMember('@AccountsReceivable:SalesId')]
    public SalesIdBase parmSalesId(SalesIdBase _salesId = salesId)
    {
        salesId = _salesId;
        return salesId;
    }

    [DataMember('InvoiceDate'), BusinessEventsDataMember('@AccountsReceivable:InvoiceDate')]
    public TransDate parmInvoiceDate(TransDate _invoiceDate = invoiceDate)
    {
        invoiceDate = _invoiceDate;
        return invoiceDate;
    }

    [DataMember('InvoiceDueDate'), BusinessEventsDataMember('@AccountsReceivable:InvoiceDueDate')]
    public DueDate parmInvoiceDueDate(DueDate _invoiceDueDate = invoiceDueDate)
    {
        invoiceDueDate = _invoiceDueDate;
        return invoiceDueDate;
    }

    [DataMember('InvoiceAmount'), BusinessEventsDataMember('@AccountsReceivable:InvoiceAmount')]
    public AmountMST parmInvoiceAmount(AmountMST _invoiceAmount = invoiceAmount)
    {
        invoiceAmount = _invoiceAmount;
        return invoiceAmount;
    }

    [DataMember('InvoiceTaxAmount'), BusinessEventsDataMember('@AccountsReceivable:InvoiceTaxAmount')]
    public TaxAmount parmInvoiceTaxAmount(TaxAmount _invoiceTaxAmount = invoiceTaxAmount)
    {
        invoiceTaxAmount = _invoiceTaxAmount;
        return invoiceTaxAmount;
    }

    [DataMember('LegalEntityDataAreaId'), BusinessEventsDataMember('@AccountsReceivable:LegalEntityDataAreaId')]
    public LegalEntityDataAreaId parmLegalEntity(LegalEntityDataAreaId _legalEntity = legalEntity)
    {
        legalEntity = _legalEntity;
        return legalEntity;
    }
}

In some cases, population of the data contract's internal state requires that you implement additional retrieval methods. These retrieval methods should be implemented as private methods, and they should be called from the initialize method.

Here is the complete implementation of the “Sales order invoice posted” business event contract.
public CustInvoiceId parmInvoiceId(CustInvoiceId _invoiceId = invoiceId)
{
    invoiceId = _invoiceId;
    return invoiceId;
}

public SalesIdBase parmSaleOrderId(SalesIdBase _salesId = salesId)
{
    salesId = _salesId;
    return salesId;
}

public TransDate parmInvoiceDate(TransDate _invoiceDate = invoiceDate)
{
    invoiceDate = _invoiceDate;
    return invoiceDate;
}

public DueDate parmInvoiceDueDate(DueDate _invoiceDueDate = invoiceDueDate)
{
    invoiceDueDate = _invoiceDueDate;
    return invoiceDueDate;
}

public AmountMST parmInvoiceAmount(AmountMST _invoiceAmount = invoiceAmount)
{
    invoiceAmount = _invoiceAmount;
    return invoiceAmount;
}

public TaxAmount parmInvoiceTaxAmount(TaxAmount _invoiceTaxAmount = invoiceTaxAmount)
{
    invoiceTaxAmount = _invoiceTaxAmount;
    return invoiceTaxAmount;
}

public LegalEntityDataAreaId parmLegalEntity(LegalEntityDataAreaId _legalEntity = legalEntity)
{
    legalEntity = _legalEntity;
    return legalEntity;
}

Sending a business event

You must modify application code so that it sends the business event at the appropriate point. Often, you can use a common point in a framework. Documents that extend SourceDocument have a common point for creating and sending a business event. For more information, see the Source document framework support section later in this topic.

Other frameworks also provide common points for sending business events. For example, the CustVendVoucher class hierarchy in the Application Object Tree (AOT) has a post method that is used to send business events that are related to posting customer or vendor vouchers. Overrides of the base class implementation provide specialization of the logic for sending business events. For an example, see CustVoucher.createBusinessEvent or VendVoucher.createBusinessEvent in the AOT.

The sending of a business event is linked to the commit of the underlying transaction. If the underlying transaction is aborted, the business event won’t be sent. Therefore, applications can send the business event at the point where the payload information is available.
if (BusinessEventsConfigurationReader::isBusinessEventEnabled(classStr(CollectionStatusUpdatedBusinessEvent)))
{
    while select dispute
        where dispute.Status == CustVendDisputeStatus::PromiseToPay
            && dispute.FollowUpDate _currentDate
        exists join custTrans
            where custTrans.RecId == dispute.CustTrans
                && !custTrans.Closed
        exists join _tmpCustAging
            where _tmpCustAging.AccountNum == custTrans.AccountNum
    {
        CollectionStatusUpdatedBusinessEvent::newFromCustDispute(dispute).send();
    }
}

Source document framework support

The source document framework supports sending business events automatically as part of the transition from an in-process state to a completed state for the document. To take advantage of this capability, documents that extend the source document framework must implement an extension of the SourceDocumentStateInProcess.getBusinessEvent method to create and return the correct BusinessEventsBase extension type.

Extending a business event payload

You might want to publish additional information as part of the payload of a business event. To send this additional information, you must extend the business event's standard payload.

Example scenario

This example shows how to extend the CustFreeTextInvoicePostedBusinessEventContract class so that it includes a customer classification. This customer classification is an industry-based custom classification.

Step 1: Create an extended business event contract

Create a contract that consists of the standard business event contract plus any additional information that must be included in the payload.

[DataContract]
public class CustFreeTextInvoicePostedBusinessEventExtendedContract extends BusinessEventsContract
{
    // standard contract
    // contract extensions
    private str customerClassification;
}

Step 2: Create an initialize method

Create an initialize method that initializes the value of the private contract.
private void initialize(CustFreeTextInvoicePostedBusinessEventContract _custFreeTextInvoicePostedBusinessEventContract) {
}

Step 3: Create a static newFrom method
Create a static newFrom method that takes the standard contract as an argument and calls the initialize method.

    var contract = new CustFreeTextInvoicePostedBusinessEventExtendedContract();
    contract.initialize(_custFreeTextInvoicePostedBusinessEventContract);
    return contract;
}

Step 4: Map parm methods
Copy the parm methods from the standard data contract, and modify each method so that it gets and sets values in the class's standard contract instance.

[DataMember('InvoiceAccount')]
public CustInvoiceAccount parmInvoiceAccount(CustInvoiceAccount _invoiceAccount = custFreeTextInvoicePostedBusinessEventContract.parmInvoiceAccount()) {
    return custFreeTextInvoicePostedBusinessEventContract.parmInvoiceAccount(_invoiceAccount);
}

[DataMember('InvoiceId')]
public CustInvoiceId parmInvoiceId(CustInvoiceId _invoiceId = custFreeTextInvoicePostedBusinessEventContract.parmInvoiceId()) {
    return custFreeTextInvoicePostedBusinessEventContract.parmInvoiceId(_invoiceId);
}

Step 5: Add parm methods for additional payload data

[DataMember('CustomerClassification')]
public CustomerClassification parmCustomerClassification(CustomerClassification _customerClassification = customerClassification) {
    customerClassification = _customerClassification;
    return customerClassification;
}

Here is the complete implementation of the extended business contract.

[DataContract]
public class CustFreeTextInvoicePostedBusinessEventExtendedContract extends BusinessEventsContract {
    // standard contract
    // contract extensions
    private str customerClassification;
        var contract = new CustFreeTextInvoicePostedBusinessEventExtendedContract();
        contract.initialize(custFreeTextInvoicePostedBusinessEventContract);
        return contract;
    }
    public CustInvoiceAccount parmInvoiceAccount(CustInvoiceAccount _invoiceAccount) {
        return custFreeTextInvoicePostedBusinessEventContract.parmInvoiceAccount(_invoiceAccount);
    }
    public CustInvoiceId parmInvoiceId(CustInvoiceId _invoiceId) {
        return custFreeTextInvoicePostedBusinessEventContract.parmInvoiceId(_invoiceId);
    }
    public CustomerClassification parmCustomerClassification(CustomerClassification _customerClassification) {
        customerClassification = _customerClassification;
        return customerClassification;
    }
}
{
    var contract = new CustFreeTextInvoicePostedBusinessEventExtendedContract();
    contract.initialize(_custFreeTextInvoicePostedBusinessEventContract);
    return contract;
}
private void initialize(CustFreeTextInvoicePostedBusinessEventContract custFreeTextInvoicePostedBusinessEventContract)
{
}
private void new()
{
}

[DataMember('InvoiceAccount')]
public CustInvoiceAccount parmInvoiceAccount(CustInvoiceAccount _invoiceAccount = custFreeTextInvoicePostedBusinessEventContract.parmInvoiceAccount())
{
    return custFreeTextInvoicePostedBusinessEventContract.parmInvoiceAccount(_invoiceAccount);
}

[DataMember('InvoiceId')]
public CustInvoiceId parmInvoiceId(CustInvoiceId _invoiceId = custFreeTextInvoicePostedBusinessEventContract.parmInvoiceId())
{
    return custFreeTextInvoicePostedBusinessEventContract.parmInvoiceId(_invoiceId);
}

[DataMember('InvoiceDate')]
public TransDate parmInvoiceDate(TransDate _invoiceDate = custFreeTextInvoicePostedBusinessEventContract.parmInvoiceDate())
{
    return custFreeTextInvoicePostedBusinessEventContract.parmInvoiceDate(_invoiceDate);
}

[DataMember('InvoiceDueDate')]
public DueDate parmInvoiceDueDate(DueDate _invoiceDueDate = custFreeTextInvoicePostedBusinessEventContract.parmInvoiceDueDate())
{
    return custFreeTextInvoicePostedBusinessEventContract.parmInvoiceDueDate(_invoiceDueDate);
}

[DataMember('InvoiceAmountInAccountingCurrency')]
public AmountMST parmInvoiceAmount(AmountMST _invoiceAmount = custFreeTextInvoicePostedBusinessEventContract.parmInvoiceAmount())
{
    return custFreeTextInvoicePostedBusinessEventContract.parmInvoiceAmount(_invoiceAmount);
}

[DataMember('InvoiceTaxAmount')]
{
    return custFreeTextInvoicePostedBusinessEventContract.parmInvoiceTaxAmount(_invoiceTaxAmount);
}

[DataMember('LegalEntity')]
public LegalEntityDataAreaId parmLegalEntity(LegalEntityDataAreaId _legalEntity = custFreeTextInvoicePostedBusinessEventContract.parmLegalEntity())
{
    return custFreeTextInvoicePostedBusinessEventContract.parmLegalEntity(_legalEntity);
}

// contract extensions
[DataMember('CustomerClassification')]
public CustomerClassification parmCustomerClassification(CustomerClassification _customerClassification = customerClassification)
Step 6: Wrap the buildContract method

Provide a build contract implementation that calls `next` to load the standard business event contract and populates any payload extensions. Here is the complete class.

```java
[ExtensionOf(classStr(CustFreeTextInvoicePostedBusinessEvent))]
public final class FreeTextInvoicePostedBusinessEventContract_Extension
{
    public BusinessEventsContract buildContract()
    {
        var businessEventContract =
            CustFreeTextInvoicePostedBusinessEventExtendedContract::newFromCustFreeTextInvoicePostedBusinessEventContra
t(next
                buildContract());
        businessEventContract.parmCustomerClassification(CustomerClassifier::deriveCustomerClassification(businessEv
entContract.parmInvoiceAccount()));
        return businessEventContract;
    }
}
```

Extending filters so that they have custom fields (if the middleware supports this extension)

Some middleware systems allow for filtering of the events. For example, Microsoft Azure Service Bus has a property bag that can be populated with key-value pairs. These key-value pairs can be used to filter events when reading from the Service Bus Queue or Topic. Additionally, Azure Event Grid has filterable message properties such as `Subject`, `Event Type`, and `ID`. To support these various properties for the different systems, the business events framework uses a concept that is named `payload context`. This concept can be extended so that it includes custom fields that the different eventing systems can use for filtering.

**Payload context**

The business events framework supports the payload context concept. Payload context provides a way to decorate messages that the framework sends with context about the payload. In some scenarios, additional context might be required when messages are sent to endpoints. Therefore, the framework has hookpoints where the context can be overwritten and the adapters can be customized.

**Step 1: Add a custom payload context**

A custom payload context must be extended from the `BusinessEventsCommitLogPayloadContext` class.

```java
class CustomCommitLogPayloadContext extends BusinessEventsCommitLogPayloadContext
{
    private utcdatetime eventTime;
    public utcdatetime parmEventTime(utcdatetime _eventTime = eventTime)
    {
        eventTime = _eventTime;
        return eventTime;
    }
}
```

**Step 2: Construct the custom payload context**

A Chain of Command (CoC) extension must be written for the `BusinessEventsSender.buildPayloadContext`
Step 3: Consume the custom payload context from an adapter

Adapters that consume payload context are written in such a way that they expose CoC methods that allow for the consumption of new payload contexts. The following example shows what this step looks like for the Service Bus adapter. This adapter has a generic property bag that Service Bus consumers can filter on.

The BusinessEventsServiceBusAdapter class has the CoC method that is named addProperties.

Adding a custom endpoint type

The business events framework supports the addition of new endpoint types to the out-of-box endpoint types. This section provides an example that shows how to add new custom endpoint types.

Step 1: Add a new endpoint type

Each endpoint type is represented by the BusinessEventsEndpointType enum. The first step in the process of adding a new endpoint is to extend this enum, as shown in the following illustration.
Step 2: Add a new endpoint table to the hierarchy

All endpoint data is stored in a hierarchy table. The root of this table is the BusinessEventsEndpoint table. A new endpoint table must extend this root table by setting the Support Inheritance property to Yes and the Extends property to "BusinessEventsEndpoint" (or any other endpoint in the BusinessEventsEndpoint hierarchy).

The new table then holds the definition of the custom fields that are required to initialize and communicate with the endpoint in code. To help avoid conflict, you should qualify field names to the specific endpoint where they belong. For example, two endpoints can have the concept of a URL field. To distinguish the fields, names should be specific to the custom endpoint. For example, name the field for the custom endpoint CustomURL.

Step 3: Add a new endpoint adapter class that implements the IBusinessEventsEndpoint interface

The new endpoint adapter class must implement the IBusinessEventsEndpoint interface. It must also be decorated with the BusinessEventsEndpointAttribute attribute.

```java
[BusinessEventsEndpoint(BusinessEventsEndpointType::CustomEndpoint)]
public class CustomEndpointAdapter implements IBusinessEventsEndpoint
{

```

The initialize method should be implemented to check the type of the BusinessEventsEndpoint buffer that is passed in, and then, if the buffer is of the correct type, initialize it, as shown in the following example.
if (!_endpoint is CustomBusinessEventsEndpoint)
{
    BusinessEventsEndpointManager::logUnknownEndpointRecord(tableStr(CustomBusinessEventsEndpoint),
        _endpoint.RecId);
}

CustomBusinessEventsEndpoint customBusinessEventsEndpoint = _endpoint as
CustomBusinessEventsEndpoint;
customField = customBusinessEventsEndpoint.CustomField;
if (!customField)
{
    throw warning(strFmt("@BusinessEvents:MissingAdapterConstructorParameter",
        classStr(CustomEndpointAdapter), varStr(customField)));
}

Step 4: Extend the EndpointConfiguration form

Add a new group control under
FormDesign/BusinessEventsEndpointConfigurationGroup/EndpointFieldsGroup/ to hold your custom field input.

The custom field input should be bound to the new table and field that you created in the previous step. Create a
class extension to extend the getConcreteType and showOtherFields methods of
BusinessEventsEndpointConfiguration form, as shown in the following example.
```java
final public class CustomBusinessEventsEndpointConfiguration_Extension {
    public TableName getConcreteTableType(BusinessEventsEndpointType _endpointType) {
        TableName tableName = next getConcreteTableType(_endpointType);
        if (_endpointType == BusinessEventsEndpointType::CustomEndpoint) {
            tableName = tableStr(CustomBusinessEventsEndpoint);
        }
        return tableName;
    }

    public void showOtherFields() {
        next showOtherFields();
        BusinessEventsEndpointType selection = any2Enum(EndpointTypeSelection.selection());
        if (selection == BusinessEventsEndpointType::CustomEndpoint) {
            this.control(this.controlId(formControlStr(BusinessEventsEndpointConfiguration, CustomFields))).visible(true);
        }
    }
}
```

Adding human-readable data fields to the payload

This feature is available in Platform update 30 and later.

The serialization of business events uses FormJsonSerializer to serialize objects in the data contract. FormJsonSerializer can format `UtcDateTime` values in the ISO 8601 date and time format. This format is human-readable when the payload of a business event is viewed. For example, a `UtcDateTime` value can now be formatted as "2007-12-05T14:30Z" instead of "/Date(1196865000000)". In the "/Date(N)" format, N is the number of milliseconds that have passed since January 1, 1970, UTC+0. The ISO format is more often understood by tools that parse JavaScript Object Notation (JSON).

To get the human-readable format, use the extended data type (EDT) that is named `DateTimeIso8601` as the type of the value in the data contract. Alternatively, use an EDT that is derived from the `DateTimeIso8601` EDT.

```java
[DataMember("TestIsoEdtUtcDateTime")]
public DateTime Iso8601 testIsoEdtUtcDateTime(DateTime Iso8601 _value = this._testIsoDateTime) {
    if (!prmIsDefault(_value)) {
        this._testIsoDateTime = _value;
    }
    return this._testIsoDateTime;
}
```
This topic provides tips for troubleshooting issues that involve business events.

<table>
<thead>
<tr>
<th>ISSUE</th>
<th>POSSIBLE RESOLUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unable to find navigation to business events in the <strong>System parameters</strong> form</td>
<td>You can access business events by going to <strong>System administration &gt; Set up &gt; Business events</strong>. This change was made when business events was made generally available in Platform update 26.</td>
</tr>
<tr>
<td><strong>Error:</strong> Unable to construct endpoint. Exception message: Error retrieving secret '{KeyValueSecretName}' from key vault 'https://{KeyVaultName}.vault.azure.net/': AADSTS700016: Application with identifier '7e28cb03-dc28-43b5-b129-e13dcf4b1fb' was not found in the directory 'ee3fe5c6-26af-42b1-9acf-5ee38e6e9d6e'.</td>
<td>This can happen if the application has not been installed by the administrator of the tenant or consented to by any user in the tenant. It's likely that you may have sent your authentication request to the wrong tenant.</td>
</tr>
<tr>
<td><strong>Error:</strong> Trace ID: 19dc9946-45b6-4335-9676-6a133db4000 Correlation ID: ecbc8a80-9d0-41ec-9c8f-d334d050bd64 Timestamp: 2019-02-06 23:27:06Z</td>
<td>This error typically means that the value in the <strong>Azure Active Directory Application ID</strong> field is incorrect. Check the <strong>Azure Active Directory Application ID</strong> value in the customers Azure portal in <strong>Azure Active Directory &gt; App Registration</strong>.</td>
</tr>
<tr>
<td><strong>Error:</strong> Unable to construct endpoint. Exception message: Error retrieving secret '{KeyValueSecretName}' from key vault 'https://{KeyVaultName}.vault.azure.net/': An error occurred while sending the request.</td>
<td>This error is likely due to an incorrect value in the <strong>Key Vault DNS Name</strong> field. To resolve this, go to the customer's Azure portal and open the key vault object. In the <strong>Overview</strong> section, check the <strong>Key Vault DNS Name</strong> value.</td>
</tr>
<tr>
<td><strong>Error:</strong> Unable to send test event to endpoint. Exception message: 40103: Invalid authorization token signature, Resource: sb://{ServiceBusName}.servicebus.windows.net/{QueueName}. TrackingId:cd0e8caa-1717-4f97-b837-4cd7eda99a4_G13, SystemTracker: [ServiceBusName].servicebus.windows.net:[QueueName], Timestamp:2019-02-06T23:36:54</td>
<td>The value in the customer's Key Vault Secret is likely incorrect. Check the <strong>Key Vault Secret</strong> value and make sure that it is correct for the endpoint type.</td>
</tr>
<tr>
<td><strong>Error:</strong> Unable to construct endpoint. Exception message: Error retrieving secret '{KeyValueSecretName}' from key vault 'https://{KeyVaultName}.vault.azure.net/': Access denied</td>
<td>This is likely due to the <strong>Azure Active Directory Application ID</strong> not having the appropriate permissions in the Key Vault. To resolve this, go to the Azure portal and open the <strong>Key Vault</strong> object. Go to <strong>Access Policies</strong> and add the AAD application with Key, Secret, and Certificate Management template.</td>
</tr>
<tr>
<td><strong>Error:</strong> Unable to send test event to endpoint. Exception message: 40400: Endpoint not found., Resource: sb://{ServiceBusName}.servicebus.windows.net/{QueueName}.</td>
<td>This issue is likely a result of the queue/topic/hub name being incorrect. Check the <strong>Name</strong> field by going to service bus in the Azure portal and reviewing the <strong>Topic</strong> or <strong>Queue</strong> name. If it is an Event Hub, go to the <strong>Event Hub</strong> object in Azure and validate the <strong>Hub</strong> name.</td>
</tr>
<tr>
<td><strong>Error:</strong> Unable to send test event to endpoint. Exception message: An error occurred while sending the request.</td>
<td>This is likely due to an incorrect endpoint value specified in the <strong>Endpoint URL</strong> field. Go to the <strong>Event Grid</strong> object in the Azure portal and open the <strong>Event Grid</strong>. In the <strong>Overview</strong> section, this value will be the <strong>Topic Endpoint</strong>.</td>
</tr>
<tr>
<td>ISSUE</td>
<td>POSSIBLE RESOLUTION</td>
</tr>
<tr>
<td>-------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Business events do not show up in the Business event catalog in Finance and Operations apps.</td>
<td>The Business event catalog is built during the full database sync. As a result, there are some use cases, as noted below, where a manual refresh of the catalog is needed in order to see the new business events. A manual refresh can be invoked from the catalog by going to Manage &gt; Rebuild business events catalog. When you are implementing business events in Visual Studio you may not see the newly-coded business event in the catalog. When new workflows are configured, such as the workflow elements or steps, events might not show up in the business events catalog. In other situations, when you don’t see certain business events, doing a manual refresh should resolve the issue.</td>
</tr>
<tr>
<td>Finance and Operations business events are not available in the Catalog Assignment tab of the virtual entity solution in the Power Platform maker portal.</td>
<td>Business events that are available in the Business event catalog in Finance and Operations apps should sync automatically to the virtual entity solution in the Power Platform portal. If the business events are not available in the Power Platform portal automatically, a sync can be triggered manually by using the Rebuild business event catalog action on the Manage menu of the Business event catalog page.</td>
</tr>
<tr>
<td>- 0 is an invalid bundle size - Business events are not getting triggered - Microsoft Flow is not getting triggered by business events - Unable to configure business event because it has reached the limit of 0 configured endpoints</td>
<td>One of the reasons why this issue can occur is if certain parameters are not set as expected in the BusinessEventsParameters table. This is due to an update in which some of the business events parameters not being set correctly. In a non-production environment, you must update the parameters in System administration &gt; Setup to set retry count = 3; End points allowed per event = 10 and wait time = 1000. After this update, restart the batch service and run IISReset to pick up the latest values. If business events still do not trigger, then the dedicated capacity is not working to process business events. A manual batch job must be scheduled to process business events, which can be enabled from the Business events parameters page in System administration &gt; Set up.</td>
</tr>
<tr>
<td>Platform update 30 compiler warning when creating custom payload context fields by augmenting via Chain of Command (CoC) the addProperties method in the adapter class. Class 'BusinessEventsServiceBusAdapter' is internal in model 'ApplicationFoundation' and cannot be extended.</td>
<td>This is a change in the compiler that prevents an internal API from being extended. This is being tracked as a bug to provide alternate ways to add custom properties. For more information, see this Yammer discussion.</td>
</tr>
<tr>
<td>Error: Unable to load one or more of the requested types. Retrieve the LoaderExceptions property for more information</td>
<td>This error message on the error tab of active business events can typically be resolved by rebuilding the catalog.</td>
</tr>
<tr>
<td>Alert business events don't trigger</td>
<td>One of the reasons why an event is not triggering could be a potential issue with alerts email functionality. Try turning off the send email option in the alert to see if that resolves the issue.</td>
</tr>
<tr>
<td>ISSUE</td>
<td>POSSIBLE RESOLUTION</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>Unable to send test event to endpoint. Exception message: The</td>
<td>Make sure the middleware is using TLS 1.2</td>
</tr>
<tr>
<td>underlying connection was closed: Could not establish trust</td>
<td></td>
</tr>
<tr>
<td>relationship for the SSL/TLS secure channel.</td>
<td></td>
</tr>
</tbody>
</table>
The following are potential use cases for business events. These use cases aren’t an exhaustive list of the potential use cases. Some of these use cases may not have been implemented yet either by Microsoft or other organizations. These use cases are meant to provide ideas and help with understanding business events.

<table>
<thead>
<tr>
<th>BUSINESS PROCESS</th>
<th>USE CASE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Procurement</td>
<td>The procurement process frequently relies on manual intervention, so automating this process should increase procurement manager productivity.</td>
<td>You can make the procurement process more efficient by alerting procurement managers when quotation requests are sent, allowing for prompt follow-up and faster engagement. The goal is to have procurement managers follow up within three days and possibly create a Microsoft Power Automate that automates this follow up.</td>
</tr>
<tr>
<td>Procurement</td>
<td>Many organizations use manual processes to communicate internally and externally about production orders. Some organizations have complex approval processes for production orders, and need quick and easy ways to review and approve orders.</td>
<td>By creating business events that are triggered when production orders are updated, you can help improve communications between the back office and the production floor. When integration is required with third-party systems for production, the business events can also be used to help simplify the integration process.</td>
</tr>
<tr>
<td>Reporting</td>
<td>Managers are not informed about newly created financial reports. As a result, managers might analyze and make decisions based on outdated data.</td>
<td>You can make the reporting process more efficient by alerting managers when financial reports are sent, allowing for prompt follow-up and faster engagement. The goal is to have managers follow up within three days and possibly create a Power Automate that automates this follow-up.</td>
</tr>
<tr>
<td>Customer master</td>
<td>When a new customer is created, a credit limit check is needed. If something could automatically trigger an API that subscribes to a credit limit, and then check the website and import-specific credit limit check fields, such as limit amount and rating. At the same time, an approval Power Automate needs to start so that the customer account can be used after approval from management.</td>
<td>This check is required by many companies, especially in the retail area. This use case would be beneficial because partners develop customizations for this purpose.</td>
</tr>
<tr>
<td>BUSINESS PROCESS</td>
<td>USE CASE</td>
<td>VALUE</td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>Month end close</td>
<td>The month-end closing schedule is a simple list that does not allow automatic actions. If functionality could be created to inform a group of people about tasks that have been completed and verified as complete, then a different group of people could start working.</td>
<td>Use real-life scenarios to enhance the currently static and difficult to use month-end closing workspace. One such use case is explained in detail in Business events in Financial Period Close.</td>
</tr>
<tr>
<td>Month end close</td>
<td>If functionality could be created to trigger a Power Automate when the status of a period is changed – either opened or closed.</td>
<td>Users are not automatically informed when new periods are opened after the month-end close is completed and they are allowed to start recording transactions in the new period.</td>
</tr>
<tr>
<td>Vendor master</td>
<td>If you are using the supplier (vendor) collaboration workspace, there is no automated way to inform the supplier that a new record has been created or existing records have been updated. This is an issue with requests for quotation and purchase orders. This means that the supplier would need to review the supplier collaboration workspace in case there are issues.</td>
<td>Using business events, the supplier portal functionality will provide improved productivity and efficiency by removing the need for additional correspondence between the supplier and the application. This is a true collaboration and will lead to supply chain efficiency. Without this functionality, the organization must follow up on all new requests or updates with phone calls or emails to the suppliers. This improves the supplier collaboration workspace.</td>
</tr>
<tr>
<td>Vendor master</td>
<td>Many organizations use manual, offline, and paper-based processes to manage vendors. Additionally, in many organizations, there are “gate keepers” that manage the master data. When updates occur to vendor master data, communication throughout the organization is limited or complex.</td>
<td>By integrating with Power Automate and creating business events for actions and events that regularly occur with vendor master data, you can help improve the overall vendor onboarding process. You can also help improve internal communications within your organization and automate business processes with your trade partners.</td>
</tr>
<tr>
<td>BUSINESS PROCESS</td>
<td>USE CASE</td>
<td>VALUE</td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>Sales quotation</td>
<td>A quotation is placed for a personalized product. After the quotation is won this needs to become a real product. Requests are sent to a Data team (using PowerApps or Office) to create the item. After the item has been manually created, the item on the line is updated and the quote triggers the creation of a sales order. Currently, when the item is created, the approach is to wait for the data integrator to copy it to Dataverse. Monitors are set to find new entries on Dataverse and compare the unique references to any open numbers. Ideally, an item created business event exists, which is extended so that it has the unique identifier. This event is then subscribed to (in Power Automate or PowerApps) and immediately updates the quote line.</td>
<td>Using business events, the business would be able to update the won quote with the new product when the new product is created. This removes possible delays, manual updates, and errors.</td>
</tr>
<tr>
<td>Sales orders</td>
<td>Shipment for sales orders starts in a different system to enable logistics balancing of third-party haulers, production of customs, and delivery documentation. This pushes the data to a major transport company. Upon picking or shipment of a sales order, the line details are the event that triggers the external system to analyze and determine the details of the utilized shipment. This pushing event depends on the ability for a business action to update the application. This could be that the order is picked and then the shipment information pushed to the application, or it is marked as shipped as it is pushed externally.</td>
<td>This allows businesses with third-party transport companies to send shipping information to their vendors. This approach puts the responsibility of dispatch utilization on the external vendors and simplifies the setup.</td>
</tr>
<tr>
<td>Sales orders</td>
<td>Picking for sales orders can be performed in different systems, whether it's a separate internal warehousing system or a third-party location that is not part of the application. Upon confirmation or picking of a sales order, the line details are the event that triggers the external system to analyze and determine the picking priority and utilization.</td>
<td>This allows businesses with third-party warehouses to send the picking information to the system. This can also be used if the warehouse has automated picking machines that determine what is required. This approach puts the responsibility of prioritizing picking and utilization on the external system and simplifies the setup in the application.</td>
</tr>
<tr>
<td>BUSINESS PROCESS</td>
<td>USE CASE</td>
<td>VALUE</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Sales orders</td>
<td>Organizations are looking for ways to improve customer service, and to streamline and automate sales processes. These processes might involve several people, require many manual steps, or involve several third-party solutions. The email notification profile feature in Finance and Operations has limited email formatting capabilities.</td>
<td>By integrating with Power Automate and creating business events for actions that are taken on sales orders, you can help improve the overall sales process, help increase customer satisfaction, and automate collaboration between sales and operations.</td>
</tr>
<tr>
<td>Batch processing</td>
<td>Scheduling batch jobs to process polling logic can constrain resources. To realize near real-time processing, it is compelling to schedule the batch to run as fast as it can in Finance and Operations, which typically ends up being every few minutes. These batch jobs are typically data import/export jobs that look for data to be processed. However, if data is not available the batch job processes empty cycles, which can consume system resources.</td>
<td>Using business events, the polling use case can be redesigned to be asynchronous if it is triggered by the business event. Data will be processed only when it is available. The business logic that makes the data available triggers the business event, which can then be used to start the data processing job/logic. This can save thousands of batch executions from running empty cycles and wasting system resources.</td>
</tr>
<tr>
<td>Production orders</td>
<td>The production scheduling process often relies on external systems or optimization engines to finalize the schedule. Automation of the process and an ability to integrate with external systems when a production order is scheduled can help increase productivity and decrease integration costs.</td>
<td>You can improve the integration and automation capabilities by creating a business event when a production order is scheduled. The information can be communicated to downstream systems, such as Microsoft Dynamics 365 Field Service for Internet of Things (IoT) data exchange, third-party manufacturing execution systems (MESs), or a scheduling optimization engine.</td>
</tr>
<tr>
<td>Production orders</td>
<td>Many organizations use third-party MESs to control and manage their machinery on the production floor. Integration with these systems is often complex. Additionally, communication between the production floor and the back office can be difficult.</td>
<td>By creating business events that are triggered when production orders are updated, you can help improve communications between the back office and the production floor. When integration is required with third-party systems for production, the business events can also be used to help simplify the integration process.</td>
</tr>
<tr>
<td>BUSINESS PROCESS</td>
<td>USE CASE</td>
<td>VALUE</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Case management</td>
<td>Case management is a platform feature that lets you track and manage various requests and issues that are internally or externally reported or requested. Typically, each type of case has a different process, and various contributors to the case might be involved throughout the case’s lifecycle. The processes can be manual, and communication can often be delayed. Sometimes, contributors to the cases or stakeholders in them might not be users of the system.</td>
<td>By using Power Automate when cases are updated, you can streamline and automate the business processes that are related to various case types. You can also automate communication and implement approval processes when necessary, depending on your organizational requirements.</td>
</tr>
<tr>
<td>Transportation management</td>
<td>Many organizations have separate and disconnected systems for managing transportation. Additionally, the staff that is responsible for arranging transportation is typically not the same staff that creates and confirms orders, or the same staff that carries out the work to ship or receive orders. This situation can lead to a breakdown or lack of communication, and can require manual processes to notify various departments when an order is ready for the next step in the process. In many cases, third-party shipping software is used, and data is entered into many systems or tracked in paper logs.</td>
<td>By using business events together with Power Automate, you can streamline the overall transportation management process and help improve communications between departments.</td>
</tr>
</tbody>
</table>
This topic explains how to configure and consume a business event from a Microsoft Power Automate endpoint by using the **When a Business Event occurs** trigger of the Finance and Operations connector.

This topic shows how to perform the following tasks:

- Create a new flow in Power Automate.
- Trigger a business event.

The steps in this topic show how use the Finance and Operations connector. However, they can also be applied to the process of creating flows in Power Automate for Finance and Operations apps business events and data events in the Microsoft Dataverse connector. For more information about Finance and Operations apps business events and data events that have the **When an action is performed** and **When a row is added, modified or deleted** triggers in the Dataverse connector, see [Business events in Microsoft Power Automate](#).

## Create a new flow in Power Automate

1. Sign in to Power Automate portal.
2. Select an existing environment where you have the permissions needed to create a Power Automate resource. The default environment is open to all companies.
3. Select **New > Create from blank**.
4. Search for [Dynamics 365 for Finance and Operations](#) and select the connector.
5. You will notice a trigger named **When a Business Event occurs**. Select this trigger.
6. Select your environment instance, category, event name, and legal entity.

### TIP

Take advantage of the auto-complete that Power Automate provides by entering only part of the environment instance URL or part of the event name.

7. Select the **New Step** button to add a new action.
8. Search for the **Parse JSON** data operation. This step is needed to parse the message with the schema of the data contract.
9. Select the content field of **Parse Json** action, then the **Body** output from the previous step should appear as an option. Select **Body**.

10. Enter the schema of the contract. Because the app provides only a sample payload you can use the Power Automate capability to generate a schema from a payload. Select an event in the catalog (for example, Customer Payment) and select the **Download schema** link. This will download a text file. Open the text file and copy the content.

11. Go Back to Power Automate and select the **Use sample payload to generate schema** link. Paste your text file content and select **Done**.
12. Depending on the quality of your sample payload, your generator will not be able to distinguish between an integer and a real number. This is true if the real number is provided as a whole number in the sample payload. Review your generated schema and check if you need to change an “integer” into “number”. (In JSON, a “number” data type means real number).

13. Choose another final action to consume the business event content. For instance, you can send an email (or post a text message to Teams) to notify the customer about payment details. Search for the Send email action, then sign in to your Microsoft 365 account.

14. Fill in the message with the required fields.

15. Save the flow.

Trigger a Business Event
Power Automate can configure the application automatically for you. After you save your flow, it creates an endpoint, then it activates the business event for you. There is no remaining configuration step apart from verifying that the endpoint has been correctly configured before triggering an event.

1. Sign in to the client.

2. Go to System Administration > Setup > Business Events.

3. Select Endpoints.

4. Verify that a new endpoint has been created with a GUID appended in the name.

5. If you check the Active events tab, you can also verify that “Payment Posted” is activated for legal entity GBSI.

6. The final step is to trigger the business event of a posted customer payment and check whether the flow runs and you receive an email with customer payment details.

Troubleshooting a flow

Here are some troubleshooting suggestions:

- Power Automate provides a full history of runs to help determine what might be wrong with a failing flow.
- When reviewing a failed run, carefully review the inputs and outputs of trigger and action blocks.
- After changes have been made to the flow, go to the latest run or a particular run, and Resubmit the inputs to run the flow again.
This topic explains how to configure a Microsoft Azure Event Grid endpoint, and how to consume a business event from Event Grid.

**Scenario overview**

Security best practices recommend that you store connection strings outside applications, in an Azure Key Vault drive, and that you give applications the correct access to the key vault keys, secrets, or certificates.

Here are two of the many benefits of this approach:

- Someone who gets access to the application database won’t be able to get the third-party connection string.
- Maintenance is easier, especially when multiple applications access the same resources, because you must update connection strings in only one place.

Here is an overview of the procedures that you must complete:

1. Create a new event grid topic.
2. Create a new key vault to store the key for the event grid topic.
3. Register an Azure app that has permission to access the key vault.
4. Configure the parameters of the endpoint.
5. Consume the business event.

**Procedure 1: Create a new event grid topic**

1. Sign in to the Azure portal.
2. Select All services > Integration > Event Grid Topics.
3. Select Add to create a new event grid topic. Set the parameters, and then select Create. You can create a new resource group as a container for your lab, or you can use an existing resource group.
4. After deployment is completed, select the new event grid. On the property blade, select Overview, and make a note of the Topic Endpoint value. You will need this value later.
5. Back on the property blade, select Access keys, and copy the Key 1 value. You will need this value when you configure the key vault in the next procedure.
Procedure 2: Create a key vault

In this procedure, you will create a key vault to store the key that you copied in the previous procedure. A key vault is a secure drive that is used to store keys, secrets, and certificates. Instead of storing the connection string, a more typical and more secure approach is to store it in a key vault. You can then register a new application with Azure Active Directory (Azure AD) and grant it the right to retrieve the secret from the key vault.

1. In the Azure portal, select All services > Security > Key vaults.

2. Create a new key vault in your resource group and set the default parameters.

3. Select Overview, then copy and save the DNS Name value for the key vault. You will use this value later.

4. Select BE-key vault > Secrets > Generate/Import. Enter a name for your secret, and paste the event grid connection string that you saved earlier.
5. Select **Create**.

**Procedure 3: Register a new application**

In this procedure, you will register a new application with Azure AD, and give it read and retrieve access to key vault secrets. The application will then use this application to retrieve event grid secrets.

1. In the Azure portal, select **All services > Security > Azure Active Directory**.
2. Select **App registrations (preview) > New registration**, and then enter a name for your application.
3. Select **Register**.
4. Select your new application, and then select **Certificates & secrets > New client secret**. Enter a name for your secret, and set the secret so that it never expires. Then select **Add**.

5. Copy and save your new secret. You will use it later.

**IMPORTANT**

Secrets are visible only one time. If you forget to copy the secret, you will have to delete it and create a new secret.

6. Select **Overview**, and copy and save the application ID. You will use this value later.
Procedure 4: Configure a Business Events endpoint

1. Sign in to the application and go to System administration > Setup > Business events.
2. Select Endpoints.
4. Select Azure Event Grid.
5. Select Next.
6. Set the required parameter values.

7. Select All services > Security > Key vaults.

8. Select the key vault that you created earlier, and then select Access policies > Add new.

9. On the Principal blade, select your new registered application. Select the check boxes for the Get and List secret permissions to retrieve key vault secrets.

10. Save your new access policy.
Procedure 5: Consume a business event

The business scenario involves sending an email message whenever a free text invoice is posted for the USMF company. The message must contain details such as the customer account number, the customer name, and the total amount of the invoice.

1. Select the business event catalog and look for free text invoice posted business event.

2. Then activate the business event for USMF company. Once activated, a test message is sent to validate the configuration and cache the connection.

3. To verify that the test message has been received, in the Azure portal, select your event grid topic, and then select Metrics. Verify that both the Published Events metric and the Unmatched Events metric show a value of at least 1. If they don’t, wait for the batch job to pick up your message.

When both metrics have a value of at least 1, you will create a new logic app to subscribe to your event grid topic.

4. Select All services > Integration > Logic Apps.
5. Create a new logic app in your resource group.

6. After your logic app resource has been created, select the option to create a blank logic app.

7. Search for Event Grid, and select the When a resource event occurs (preview) trigger.

8. Select your subscription, select Microsoft.EventGrid.Topics as the resource type, and select the name of the event grid topic that you created in procedure 1.

9. Select New Step to add a new action.

10. Search for the Parse Json data operation. This step is required so that the message can be parsed by using the provided schema for the data contract.

11. Click in the Content field of the Parse Json action. The pane that appears gives you the option form the previous trigger. You must select the Data object field of the event grid message that contains the
payload that is transmitted by Finance and Operations.

Next, you must enter the provided schema for the contract. This is only a sample payload. However, you can use a capability of Azure Logic Apps to generate a schema from a payload.

12. Select your event in the business event catalog, and then select the **Download schema** link. A text file is downloaded. Open the text file, and copy the contents.

13. Go back to Logic Apps, and select the **Use sample payload to generate schema** link. Paste the contents of the text file, and then select **Done**.

14. Depending on the quality of your sample payload, your generator won't be able to distinguish between an integer and a real value, especially if the real value is provided as a whole number in the sample payload. Review the schema that is generated, and determine whether you must change a field of the integer data type to the number data type. (In JavaScript Object Notation [JSON], the number data type represents real values.)

Next, you will select a final action, such as sending a notification email that includes customer payment.
15. Search for the send email action, and then sign in to your Microsoft 365 account.

16. Fill in the message with the required fields.

17. Save your logic app.

18. Trigger the business event by posting a customer payment. Then verify that the logic app runs, and that you receive an email that includes customer payment details.
This topic explains how to configure a Microsoft Azure Service Bus endpoint and how to consume a business event from Service Bus.

Scenario overview

Security best practices recommend that you store connection strings outside applications, in an Azure Key Vault drive, and that you give applications the correct access to the key vault keys, secrets, or certificates.

Here are two of the many benefits of this approach:

- Someone who gets access to the application database won't be able to get the third-party connection string.
- Maintenance is easier, especially when multiple applications access the same resources, because you must update connection strings in only one place.

Here is an overview of the procedures that you must complete:

1. Create a new Service Bus namespace.
2. Create a new Service Bus topic and subscription.
3. Create a new key vault to store the Service Bus key.
4. Register an Azure app that has permission to access the key vault.
5. Configure a Business Events endpoint.
6. Consume the business event.

Create a new Service Bus namespace

1. Sign in to the Azure portal.
2. Select All services > Integration > Service Bus.
3. Select Add to create a new Service Bus namespace, and set the parameters. Select the Standard pricing tier. You can create a new resource group as a container for your lab, or you can use an existing resource group.

   **NOTE**
   
   If you select the Basic pricing tier, you can create only queues. To create topics, you must select the Standard pricing tier.

4. When you've finished setting all the parameters, select Create.

Create a new Service Bus topic and subscription

1. In the Azure portal, select the Service Bus that you just created, and then create a new topic.
2. Select the new topic, and then create a new subscription that is named **BE-USMF**.

3. Go back to the blade for your Service Bus, and create a new shared access policy to send events. Only the **Send** policy is required to send events to the Service Bus topic.

4. Select the new **Send** policy, and then copy and save the **Primary Connection String** value. You will use this value later.
Create a new key vault

In this procedure, you will create a key vault to store the key that you copied in the previous procedure. A key vault is a secure drive that is used to store keys, secrets, and certificates. Instead of storing the connection string, a more typical and more secure approach is to store it in a key vault. You can then register a new application with Azure Active Directory (Azure AD) and grant it the right to retrieve the secret from the key vault.

1. In the Azure portal, select All services > Security > Key vaults.

2. Create a new key vault in your resource group and set the default parameters.

3. Select Overview, then copy and save the DNS Name value for the key vault. You will use this value later.

4. Select BE-key vault > Secrets > Generate/Import. Enter a name for your secret, and paste the Service Bus connection string that you saved earlier.
5. Select Create.

Register a new application

In this procedure, you will register a new application with Azure AD, and give it read and retrieve access to key vault secrets. Finance and Operations will then use this application to retrieve Service Bus secrets.

1. In the Azure portal, select **All services > Security > Azure Active Directory**.

2. Select **App registrations (preview) > New registration**, and enter a name for your application.

3. Select **Register**.

4. Select the new application, and then select **Certificates & secrets > New client secret**. Enter a name for your secret, and set the secret so that it never expires. Then select **Add**.

5. Copy and save your new secret. You will use it later.

**IMPORTANT**

Secrets are visible only one time. If you forget to copy the secret, you will have to delete it and create a new secret.

6. Select **Overview**, and copy and save the application ID. You will use this value later.
7. Select All services > Security > Key vaults.

8. Select the key vault that you created earlier, and then select Access policies > Add new.

9. On the Principal blade, select your new registered application. Select the check boxes for the Get and List secret permissions to retrieve key vault secrets.

10. Save your new access policy.

**Configure a Business Events endpoint**

1. Sign in to the application and go to System administration > Setup > Business events.

2. Select Endpoints.


5. Select Next.

6. Set the required parameter values.
The business scenario involves sending an email or a message to a team channel whenever a customer payment is posted for the USMF company. The message must contain details such as the customer account number, the customer name, and the amount of the payment.

1. Select the business event catalog and look for **customer payment posted** business event.
2. Activate the business event for USMF company.

After you activate a business event that uses the new Service Bus endpoint, the application sends a test message to verify that the configuration is accurate and to cache the connection.

3. To verify that the test message has been received, in the Azure portal, select your **BE-Topic** Service Bus topic, and then go into the **BE-USMF** Service Bus subscription that you created earlier. Verify that the message count for the subscription shows a value of at least 1. If it doesn’t, wait for the batch job to pick up your message.
4. Select **All services > Integration > Logic Apps**.

5. Create a new logic app in your resource group.

6. After your Logic Apps resource has been created, select the option to create a blank logic app.

7. Search for **Service Bus**, and select it.

8. Select the trigger that is named **When a message is received in a topic subscription (auto-complete)**.

   **NOTE**
   
   Auto-complete means that the message is deleted from the subscription queue after it's retrieved. Peek-lock authorizes concurrent consumers. It requires a call to the `complete` command of the Service Bus application programming interface (API) in order to delete the message.

Because Logic Apps is accessing your Service Bus for the first time, it asks for a new connection. This connection will cache connection details as a Service Bus namespace URL and credential.

9. Select your Service Bus namespace, and enter a name for the new connection.

10. Select the **RootManageSharedAccessKey** policy for your logic app, and then select **Create**.

   **NOTE**
   
   The **Send** policy can't be used here, because you want to **retrieve** messages, not send them. As a best practice, you could have created a new policy for this use case and given it **Listen** permission only.
11. Select your trigger parameters. Be sure to use the correct names for the topic and subscription that you created.

This API polls Service Bus for new messages at a configurable recurrence (by default, every three minutes). If the volume of messages is low, the API will have a cost impact for unnecessary triggers, because Logic Apps is priced per trigger call and action run. However, you can implement a push architecture that uses Azure Event Grid in the middle. Service Bus can then push events to Event Grid when there are messages in a queue or a subscription. For more information, see Azure Service Bus to Event Grid integration overview.

12. Select New step to add a new action.

13. Search for the Parse Json data operation. This step is required so that the message can be parsed by using the schema of the data contract.

The body content that is received from the Service Bus is encoded into base64 format. Therefore, you must transform it to string format before the JavaScript Object Notation (JSON) payload can be parsed.

14. Click in the Content field, and then, in the pane that appears, on the Expression tab, enter the following expression: `Base64ToString()`
15. Put the cursor between the parentheses in the expression, and then, on the Dynamic content tab, find and select the Content of the message content from the previous Service Bus trigger. Then select OK.

Next, you must enter the schema of the contract that is received from the application. The application only provides a sample payload. However, you can use a capability of Azure Logic Apps to generate a schema from a payload.

16. Select your event in the business event catalog, and then select the Download schema link. Open the text file that is downloaded, and copy the contents.

17. Go back to your Logic Apps, and select the Use sample payload to generate schema link. Paste the contents of the text file, and then select Done.

18. Depending on the quality of your sample payload, your generator won't be able to distinguish between
an integer and a real value, especially if the real value is provided as a whole number in the sample payload. Review the schema that is generated, and determine whether you must change a field of the integer data type to the number data type. (In JSON, the number data type represents real values.)

Next, you will select a final action, such as sending a notification email that includes customer payment details.

19. Search for the send email action, and then sign in to your Microsoft 365 account.

20. Fill in the message with the required fields.

21. Save your logic app.

22. Trigger the business event by posting a customer payment. Then verify that the logic app runs, and that you receive an email that includes customer payment details.
This topic explains how to configure a Microsoft Azure Service Bus Queue endpoint.

Create an Azure Service Bus Queue endpoint

1. On the Business events page, on the Endpoints tab, select New to create an endpoint.
2. In the Configure new endpoint dialog box, in the Endpoint type field, select the appropriate endpoint type. To create an endpoint to a Service Bus queue, select Azure Service Bus Queue.
3. Select Next.
4. In the Endpoint name field, enter the name of the endpoint.
5. Set up Azure Key Vault to provide the secret to the Azure messaging resource.
6. Set up the Azure Active Directory (Azure AD) application ID and application secret.
7. Back in the Configure new endpoint dialog box, in the Queue name field, enter the name that you created for the Service Bus queue in the Azure Service Bus Queue configuration in Azure.
8. In the Azure Active Directory application ID field, enter the application ID that you created in Azure AD in the Azure portal.
9. In the Azure application secret field, enter the secret value for the application.

10. In the Key Vault DNS name field, enter the Domain Name System (DNS) name from your Key Vault setup.

11. In the Key Vault secret name field, enter the secret name for the endpoint resource that must be created in Key Vault.

The Key Vault Secret value in Azure will be the Primary Connection String value for the Service Bus. You can find this value in the Service Bus that you configured, at Shared Access Policies > RootManagedSharedAccessKey.
12. Select OK.

**IMPORTANT**

The Azure application that was registered must be also added to the Key Vault setup under Access policies in the key vault. To complete this setup, select the Key, Secret & Certificate Management template, and then select the application as the principal.
This topic explains how to use Microsoft Power Automate to configure and consume a workflow business event for purchase requisition approval.

To complete this topic, you must be running Microsoft Dynamics 365 for Finance and Operations version 10.0.2 (May 2019) with platform update 26 or later.

Scenario overview

The following illustration shows the high-level process that you must configure by using Power Automate. Note the following points:

1. The application fires a business event whenever a new approval starts.
2. Power Automate trigger starts.
3. After parsing business event payload from F&O, next step is to check whether the workflow instance ID received from F&O is still alive. This is a security step in case approval has already taken place or workflow has been recalled.
4. If the check is unsuccessful, an email is sent to notify the user about a potential work item in their workspace.
5. If the check is successful, a new Power Automate approval is started.
6. Then workflow is completed by using the outcome of the approval. The outcome can be either Approve or Reject.

Exercise 1: Create a new flow

1. Sign in to the Power Automate portal.
2. Select an existing environment where you have the right to create a flow resource. The (default) environment is available to all companies.
3. Select New > Create from blank.
4. Search for Dynamics 365 for Finance and Operations, and select the connector.
5. A new trigger is created. This trigger is named When a Business Event occurs. Select it.
6. Select the environment instance that has these characteristics:
   - The category is Workflow workitem.
• The event name is **Purchase requisition review (000062) – Approve purchase requisitions**.
• Any legal entity is selected.

7. Select **New Step** to add a new action.

8. Search for the **Parse Json** data operation. This step is required so that the message can be parsed by using the schema of the data contract that the application provides.

9. Select the content field of the **Parse Json** action. The **Body** output from the previous step should appear as an option. Select **Body**.

    Next, you must enter the schema of the contract. The application provides only a sample payload. However, you can use a capability of Microsoft Flow to generate a schema from a payload.

10. Select the **000062** workflow event in the catalog, and then select the **Download schema** link. Open the text file that is downloaded, and copy the contents.

11. Go back to Power Automate, and select the **Use sample payload to generate schema** link. Paste the contents of the text file, and then select **Done**.

12. Add a new step to call a workflow action that validates whether a workflow that has the correct instance ID is running and awaiting approval.

13. Add a new condition control step to check the result of the validate action. The dynamic field won't provide the required output. Therefore, you must manually enter the following expression instead: **Body('Execute_action')?['value']**. Then select **OK**.

14. The condition control automatically creates two branches for **Yes/No** results. If the result of the validate step is **No**, an email must be sent to the user. This email notifies the user that a new task requires
attention, and that the user must sign in to the client. In order to complete this step create a new send email action within the No container and fill in the parameter with the email of the Approver from the previous step workflowuseremail and a subject and body of your choice.

NOTE
The email address that the workflow business event returns is the email address of the workflow approver. If the workflow approver user hasn't been configured in your demo environment, you can use your own email address for demo purposes.

15. If the result of the validate step is Yes, you must start a new Power Automate approval step. In the Yes container, select a new action that is named Start and wait for an approval (v2), and choose inputs as follows: Approval type: Approve/reject: first to respond title: workflowworkitemsubject output form Business event payload Assigned to: workflowuseremail output Then you can fill in the details section with as much information as needed from previous step such as workflowdocument or workflowstepinstruction. Again, you can use your own email address in the Assigned to field for demo purposes especially if the workflow approver user hasn't been configured in your demo environment.
16. Next, you must complete the workflow approval by using the outcome of the approval step. Still in the **Yes** container, add a new **Finance and Operations Execute Action** step, and choose the **WorkflowWorkItem-complete** action and the **WorkflowWorkItemInstanceID** parameter. Then fill in the rest of the parameters from the approval outputs. As a minimum the outcome section with Approval outcome and the comment section with the approver’s responses. Because the approval step can support multiple approvers, the response output is an array. Therefore, as soon as you select the output **Responses** as an input for the comment section, Power Automate automatically embeds your action in an **Apply to each** container as shown below.
Exercise 2: Trigger a business event

Power Automate can automatically configure the application for you. After you save your flow, Power Automate creates an endpoint and activates the business event. You don't have to complete any other configurations. You just have to verify that the endpoint has been correctly configured and then trigger an event.

1. Sign in to the client.
2. Go to System administration > Setup > Business events.
3. Select Business events.
4. Select Endpoints.
5. Verify that a new endpoint has been created, and that a globally unique identifier (GUID) has been appended to the name.
6. On the Active events tab, verify that Workflow workitem is activated for the USMF company.
This topic goes through a scenario where Microsoft Forms Pro is used to create a survey that can be used with business events. Specifically, in the scenario that is described here, a survey is sent to customers when a product has been shipped. The survey information is gathered by using Forms Pro.

**Prerequisites**

If you haven't used Forms Pro before, you should first read the [Forms Pro documentation](https://docs.microsoft.com/en-us/microsoft Forms) to learn how to use it.

**Scenario**

1. Create a survey. Based on the title that you enter for the survey, Forms Pro suggests survey questions.

   ![Customer Satisfaction Survey](image)

   - **Overall, how satisfied are you with our company?**
   - **Please explain your satisfaction level in the question above.**
   - **How well do our products meet your needs?**
   - **How would you rate the value for money of the product?**
   - **How responsive have we been to your questions or concerns about our products?**
   - **How long has your organization been a customer of our company?**
   - **How likely are you to continue doing business with us in the future?**
   - **How likely are you to recommend our company to a friend or colleague?**
   - **Please share any additional comments or questions:**

2. The sales order tracks the shipment. When the product has been shipped, the status of the sales order is changed to **Delivered**.
Therefore, configure an alert on the sales order, so that an alert is created whenever the value of the Status field is changed. Be sure to set the Send externally option to Yes, so that the alert will be sent out as a business event.

3. Set up the flow that will be triggered by the business event whenever the status of the sales order is updated (see the illustration in the next step). After it’s triggered, the flow will use the Forms connector to send the survey to the customer email address that is registered on the sales order.

The customer email address and other information that is required for the scenario must be in the payload of the business event. If the payload doesn’t have this data, it can be extended so that it includes the appropriate fields. For more information, see the Business events developer documentation.

4. Because Microsoft Power Automate is used to orchestrate this scenario, don’t activate the When a change based alert occurs business event in the application. Instead, set up Power Automate so that it subscribes directly to the business event.
5. After you’ve finished setting up the flow, it will be triggered and send out the survey whenever the sales order’s status is updated.

As users fill in the survey and submit it, Forms Pro shows some analytics.
100% of responders have expressed Positive sentiment.
This tutorial describes the steps that you must follow to make business events work with Microsoft Azure Event Hubs.

1. In Azure portal, create an Active Directory application registration. Make a note of the application ID.

2. Give the app, permission to the Azure Key Vault application programming interface (API).

   **API permissions**: Applications are authorized to use APIs by requesting permissions. These permissions show up during the consent process where users grant/deny access.

<table>
<thead>
<tr>
<th>API / PERMISSIONS NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Azure Key Vault (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>user_impersonation</td>
<td>Delegated</td>
<td>Have full access to the Azure Key Vault service</td>
</tr>
<tr>
<td>Microsoft Graph (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User.Read</td>
<td>Delegated</td>
<td>Sign in and read user profile</td>
</tr>
</tbody>
</table>

3. In the app registration, create an application secret. Make a note of the value.

   Credentials enable applications to identify themselves to the authentication service when receiving tokens at a web addressable location (using an HTTPS scheme). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a credential.

   **Certificates**
   Certificates can be used as secrets to prove the application's identity when requesting a token. Also can be referred to as public keys.

   No certificates have been added for this application.

   **Client secrets**
   A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>EXPIRES</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>vaultAppSecret</td>
<td>12/01/2030</td>
<td>My************</td>
</tr>
</tbody>
</table>

4. In the key vault, give permission to the new app registration.
5. In the key vault, create a new secret. The value of this secret must be the connection string to your event hub. Make a note of the value.

6. Create an endpoint configuration for the event hub. Go to System administration > Setup > Business events > Business events catalog, and then, on the Endpoints tab, select New to open the Configure new endpoint wizard.
7. In the **Endpoint type** field, select **Azure Event Hub**.

8. Select **Next**.

9. In the **Endpoint name** field, enter a name for the endpoint.

10. In the **Hub name** field, enter the name of your event hub.

11. In the **Azure Active Directory application ID** field, enter the application ID that was created earlier.

12. In the **Azure application secret** field, enter the value that was created earlier.

13. In the **Key Vault DNS name** field, enter the Domain Name System (DNS) name of your key vault. You can find this value on the **Overview** tab of the key vault configuration in the Azure portal.

14. In the **Key Vault secret name** field, enter the name from the secret that was created earlier.

15. Select **OK**.

16. You can now activate one or more business events that should be sent to this endpoint.
IMPORTANT

Before you subscribe to Finance and Operations apps business events and data events in Microsoft Dataverse, you must enable the Microsoft Power Platform integration. For information about how to enable the Microsoft Power Platform integration for a Finance and Operations apps environment, see Enabling the Power Platform integration.

You can subscribe to Finance and Operations apps business events and data events in Dataverse by registering plug-ins and software development kit (SDK) steps on the events in Dataverse. This topic describes how to use the Power Platform Tools extension for Visual Studio to register a plug-in for a Finance and Operations apps event. The subscriptions are shown together with other subscriptions in the business event catalog in Finance and Operations apps. The endpoint then works like other endpoints in the Finance and Operations apps business event catalog.

Set up your development environment

Install the Power Platform Tools extension

Power Platform Tools for Visual Studio is an extension that provides code templates for Dataverse plug-ins. It includes a Dataverse explorer that shows tables, business events, and virtual entity data events. The explorer lets you register plug-ins directly from Visual Studio. The C# code for the plug-in can then be deployed directly to Dataverse from the solution.

For information about how to install the Power Platform Tools extension, see Install Power Platform Tools.

Create a project

After you've installed the Power Platform Tools extension, create a new project.

1. Open Visual Studio 2019 or later.
2. In the Get started dialog box, select Create a new project.
3. In the Create a new project dialog box, search for Power Platform Solution Template, select it, and then select Next.
4. In the **Configure your new project** dialog box, enter a project name, select the location where you want to save the solution file, and then select **Create**.

5. In the **Configure Microsoft Power Platform Solution** dialog box, under 1. **Solution Type to Configure**, select **Start from Dataverse**.

6. In the **Power Platform Tools** dialog box, under 2. **Connect to Dataverse**, follow these steps:
   
a. In the Deployment Type field group, select the **Office 365** option.
b. Select the Display list of available organizations checkbox.
c. Select **Login**, and enter the credentials to sign in to the Dataverse environment that is linked to your Finance and Operations apps environment.
d. In the list of organizations, select the Microsoft Power Platform environment that you want to work with. Then select **Login**.
e. Select **Next**.
7. Under **Select Solution**, select the Microsoft Power Platform solution where you want to create the event subscription. If you haven’t yet created a solution, you can create one in the Power Apps maker portal by following the steps in **Create a solution**.

8. Select **Done**.

9. In the **Visual Studio Template Selection Microsoft Power Platform** dialog box, under **1. Select**
Items for Template, select Add New Templates.

10. In the Create New Items dialog box, under 2. Selected Template Projects, select the Add Plugin Project checkbox, and then select Next.

11. Under 3. Assign project Names, in the Plugins field, enter a name for the plug-in project. The name will be the name of the Visual Studio project. By default, it will also be the name of the assembly.

12. Select Done.

For more information about how to use the Power Platform Tools extension to create a project, see Quickstart:
Create a Power Platform Tools project.

Sign the assemblies
Dataverse assemblies must be signed. You can either set up a self-signed key in the solution or provide another key if you have one available. To create a self-signed key, follow these steps.

1. In Visual Studio, in Solution Explorer, select and hold (or right-click) the project name, and then select Properties.
2. On the Signing tab, select the Sign the assembly checkbox.
3. In the Choose a strong name key file field, select New.
4. Enter a name and password for the key, and then select OK.

Subscribe to a Finance and Operations apps event
After you've finished setting up the development environment, you can begin to write code. You can create a C# class library that runs business logic in Dataverse when a Finance and Operations apps business event or data event that the plug-in is subscribed to occurs.

Register a new step
1. In Visual Studio, on the View menu, select Power Platform Explorer.

   Power Platform Explorer shows a list of components from the Dataverse environment that you selected during the setup of the development environment. These components include tables, choices, and event catalogs.

2. Under the Event Catalog node, expand Finance and Operations.

   Under the Finance and Operations node, you should see a list of catalogs that are available in the Dynamics 365 ERP Virtual Entities solution in the selected Microsoft Power Platform environment. Under each catalog, you should see a list of the virtual entities that have been generated for that category in the environment, and the data events that are available for each of those virtual entities (Created, Updated, and Deleted).

   (If you don't see any catalogs under the Finance and Operations node, you might have to generate and enable the virtual entities that are required for your solution. For more information about how to generate virtual entities in a Dataverse environment, see Enable Microsoft Dataverse virtual entities. After you've enabled the required virtual entities, select Refresh in Power Platform Explorer to update the list so that it shows the entities.)

   Under each catalog, under the Global node, you should see all the Finance and Operations apps business events that have been activated for the category.

3. Select and hold (or right-click) the data event under the virtual entity that should trigger your business logic, and then select Add Plugin.
4. In the Register New Step dialog box, follow these steps:

   a. In the Class Name field, change the value to the name of the class that you want to create.
   b. Under Event Pipeline Stage of Execution, select PostOperation in the drop-down list.
   c. Under Execution Mode, select the Asynchronous option.
   d. Select Register New Step.

   **NOTE**

   The PreValidation and PreOperation execution stages and the Synchronous execution mode aren’t currently supported for virtual entities.

When the new step is registered, a new class is generated that has the base code for the plug-in. In the **ExecuteCdsPlugin** method of the new class, you can write the custom business logic in the place that is indicated by the **TODO**. You can then build your solution and deploy the plug-in to your environment.
Deploy the plug-in

You can now deploy your plug-in to the Microsoft Power Platform solution.

- In Visual Studio, in Solution Explorer, select and hold (or right-click) the project, and then select Deploy.

To verify that deployment was successful, you can view the plug-in assembly in the Power Apps maker portal. Go to the Microsoft Power Platform solution that you deployed the plug-in to, and then, in the navigation on the left, select Plug-in assemblies. To view the registered plug-in step, select Plug-in steps in the navigation.

You can also verify that the new endpoint appears correctly on the Endpoints tab of the Business events page in the Finance and Operations app, and that the new event appears on the Active events tab.

Troubleshooting

If deployment fails, you can troubleshoot the issue by turning on verbose logging.

1. In Visual Studio, on the Tools menu, select Options.
2. In the Options dialog box, under the Power Platform Tools node, select General.
3. Select Display Detailed Log Data and (Diagnostics) Capture Detailed Dataverse Communications Log.

4. Select OK.
This topic explains how to use business events in the financial period close business process to gain insights and provide internal controls.

To complete this topic, you must be running version 10.0.2 (May 2019) with Platform update 26 or later.

Scenario overview

Task management is fundamental to managing business processes across industries. Out-of-box capabilities let users manage business process tasks in a structured manner. The Financial period close workspace illustrates these capabilities by offering a central location for managing tasks in a company’s accounting period close process.

This topic looks at an organization that recently decided to explore how it can use the Financial period close workspace to track and report tasks that are associated with every period close. Performance management and traceability are some of the challenges that this organization faces in the current setup. Therefore, the organization undertook an exercise in business process transformation to identify the capabilities of the Financial period close workspace. This exercise revealed the following business requirements:

1. The ability to be notified when tasks must be started
2. The ability to attach documents
3. Record management and disposition capabilities for attachments
4. The ability for multiple approvers to approve tasks, based on predefined logic
5. Task questionnaires for audits
6. Reporting capabilities to track the current status of the period close process and do performance analysis for insights into efficiency

High-level design

To achieve the previously mentioned requirements, the organization used out-of-box capabilities of the Financial period close workspace. A gap analysis revealed that, by doing minor extensions to the workspace and the underlying data entities, the organization could achieve requirements 2, 5, and 6, and could partially achieve requirement 4. To achieve requirements 1 and 3, and parts of requirement 4, the organization chose to use Power Automate. The following illustration shows an architectural overview of the solution.
Managing attachments by using Microsoft Power Automate and SharePoint Online

Accountants view their tasks in the Financial period close workspace and start to work on them. Attachments are added to the task by using a SharePoint Online document type. SharePoint triggers in Microsoft Power Automate are used to trigger the Power Automate that is shown in the following illustration. This Power Automate updates the SharePoint metadata with metadata from the task in the Financial period close workspace. SharePoint columns were created for this purpose in the document library. A separate attachment data entity was created to hold the attachment metadata for every attachment that is added to the Financial period close workspace. Fields from the custom entity were mapped to the SharePoint Online columns in the Power Automate. When documents that use the specified document type are created in the predefined SharePoint Online library, Power Automate is triggered, obtains the metadata from the custom data entity, and updates the document's metadata columns in SharePoint Online.

Enabling internal controls by using business events and Power Automate

As accountants complete their tasks, and the tasks become ready for review, the value of the Review status custom field is updated to Ready for review. The Power Automate gets triggered by the When the change-based alert is triggered business event when this update is made. The payload of this business event contains the task name and the area name. The Power Automate uses the combination of the task name and area name, together with the value of the Review status field, to route the task through an email-based workflow that is orchestrated by Power Automate. The Power Automate waits for approval, add new comments to the task log, and updates the task in the Financial period close workspace, based on both the outcome of the approval process and related metadata. Custom data entities were built in to query and update the Financial period close workspace by using Power Automate.

Subscribing to the business event

The following example describes the general steps for subscribing to a change-based alert business event.

1. Add the connector trigger to the Power Automate app, and subscribe to the change-based alert business event.
2. Parse the business event payload.

When the business event is triggered, it triggers Power Automate. This business event contains a payload. In this step, the payload is parsed, and the required variables are initialized.

3. Retrieve the task, based on the values from the payload.

When the task is updated, the business event triggers Power Automate. At that point, after the payload has been parsed, you will know basic information about the task. In this step, the custom data entity is used to retrieve more information about the task.
4. Retrieve approvers from the Microsoft Excel file, based on the criteria.

Next, you must determine the list of approvers, so that you can send the approval request in the appropriate manner. This list is a custom Excel file in a SharePoint Online library. In this step, you query the Excel file to get the list of approvers. You also get the links to the attachments for each task, so that you can send the attachments to the approvers.

5. Prepare to send the request for approval.

In this step, you prepare Power Automate to send the approval request by using all the information that was gathered and assembled in the previous step.
6. Start the approval process.

In this step, the approval request is sent from Power Automate.
7. Process the approval action that is taken by approvers.

After the approvers receive the approval request and take action, the Power Automate is notified, and additional processing is done.

8. Update the task with the approval outcome.

Based on the outcome of the approval process, the task is updated with the result.
Conclusion

For the business requirements of the organization that is described in this topic, this solution involves minimal development and relies mostly on the Financial period close workspace, business events, SharePoint Online, and Power Automate to drive functionality. Development is restricted to the addition of fields to pages, the creation of custom data entities, and changes to page labels. Power Automate also provides greater flexibility in the approval process. Because the solution takes advantage of the various applications in the Microsoft 365 suite, internal users can use applications that they are already familiar with. Therefore, the amount of change management that is required is limited.

In conclusion, business events offer unique opportunities for extending functionality but also let you avoid extensive in-app customizations. Here are some things to consider before you start to use business events:

- Establish the security requirements of your solution. Business events honor role-based security. This behavior
can be beneficial in some use cases.

- Business events functionality continues to get enhanced. Be on the lookout for new capabilities.

Business events and Power Automate offer great opportunities for implementing low-code or no-code extensions. The important thing is that you identify opportunities where this framework can help, but that you also understand some of the limitations.
Add-ins provide a way to extend the functionality of Finance and Operations apps. All add-ins are installed and managed via the environment details page for sandbox and production environments in Microsoft Dynamics Lifecycle Services (LCS). For more information about the architecture and how to unlock this feature, see Microsoft Power Platform integration with Finance and Operations apps.

What add-ins are available?

New add-ins are made available on a regular basis. This section describes the add-ins that are currently available and provides links to more information about each.

**Planning Optimization**
The Planning Optimization Add-in for Microsoft Dynamics 365 Supply Chain Management enables master planning calculation to occur outside Dynamics 365 Supply Chain Management and the related SQL database. The benefits that are associated with the Planning Optimization functionality include improved performance and minimal impact on the SQL database during master planning runs. Quick planning runs can be done even during office hours. Therefore, planners can immediately react to demand or parameter changes. To learn more, see Planning Optimization overview.

**Inventory Visibility**
The Inventory Visibility Add-in is an independent and highly scalable microservice that enables real-time tracking of on-hand inventory. Therefore, it provides a global view of inventory visibility. To learn more, see Inventory Visibility Add-in.

**Export to Azure Data Lake**
The Export to Azure Data Lake feature is based on a microservice that exports Finance and Operations app data to Azure Data Lake and keeps the data fresh. To learn more, see Configure export to Azure Data Lake.

**IoT Intelligence**
IoT Intelligence is an add-in for Supply Chain Management. It integrates Internet of Things (IoT) signals with data in Supply Chain Management to produce actionable insights. To learn more, see IoT Intelligence home page.
By using mobile apps, you can reuse business logic and modeling. Mobile apps enable rich offline and mobile interactions, and provide an easy-to-use designer experience. Developers can create simplified forms in Microsoft Visual Studio and then design mobile apps that expose this functionality. The mobile platform makes it easy to change the forms and mobile app definitions to include customizations that are made to your cloud app.

Get started

- Getting started
- Architecture
- Page design guidelines
- Action design guidelines
- Form design requirements

Check out the following series of how-to videos that show how to create a mobile app.

- Tutorial 1: Building the sales order page
- Tutorial 2: Building the sales order details page
- Tutorial 3: Building the create new sales order action
- Tutorial 4: Adding a lookup to the create new sales order action
- Tutorial 5: Adding a lookup and hiding pages using mobile business logic

Common configurations

These topics describe some common customizations that you can add to your mobile app.

- Localize mobile workspaces
- Help secure mobile workspaces
- Set up clickable fields
- Set up mandatory fields through workspace classes
- Display item counts in a field

Client-side development

Client-side APIs are used in the business logic file, which provides an extensibility layer to the mobile workspace that allows for customization. Some things that you can access and influence through the client-side APIs include:

- Metadata
- Runtime control/page instances
- Business data
- Offline-first business behaviors
- Layout and style

The process for client-side development is described in these topics:

- Client-side design APIs overview
Business logic events overview

Client APIs

You can download a sample business logic file (with a .js file name extension) for the Reservation management workspace. Go to Dynamics365-for-Operations-mobile-FleetManagementSamples, open the business_logic folder, and locate the FM.js file.

Server-side development

Workspace attributes and classes are used to create, configure, and publish workspaces on the server. These server-side X++ APIs can be used instead of using the task recorder-based mechanism to build a workspace. Workspaces created using either mechanism can then be styled and augmented using the client-side APIs.

Server-side development is described in these topics:

- Workspace class overview
- Server APIs (X++)

You can download the sample project (with an .axpp file name extension) for the Fleet Management mobile app. Go to Dynamics365-for-Operations-mobile-FleetManagementSamples and download the FMMobileApp.axpp file.

Debugging during development

During development it can be useful to attach a debugger to get more detailed information and insight into what is happening in the background. A web debugger can be used with the client-side JavaScript logic and styling and the Visual Studio debugger can be used with the server-side X++ business logic.

Debugging the client side

Prerequisites

- Android device plus PC
- Azure-hosted development machine (so the mobile device can point to it)

Steps to debug the client side

1. On the web client that is exposed by the Azure-hosted development machine, ensure that there are mobile workspaces published for the Finance and Operations app. For information about publishing a mobile workspace, see Publish mobile workspaces.

2. Install the Android debug apk for the Finance and Operations app on an Android device:

   - One time only, allow the installation of apk files - Go to Menu > Settings > Security and then check Unknown Sources to allow the phone to install apps from sources other than the Google Play Store.
   - Uninstall the Finance and Operations app - Ensure that any previous version of the Finance and Operations app has been uninstalled.
   - Download the apk file - From the device's browser, navigate to the latest Finance and Operations Android debug apk on GitHub and click Download (or use this direct link to the file).
   - Install the Finance and Operations apk file - Confirm install of the Finance and Operations app via the apk file.
   - Run the debug Finance and Operations app on the device and sign in.

3. Connect to the device from the debugging machine.

   - On the Azure-hosted development machine or a separate PC, follow Android developer instructions to Get Started with Remote Debugging Android Devices. You can also find a wide selection of instructional videos on YouTube by searching for Chrome for Android remote debugging.

4. After you connect the debugger, find the active tab on your device. You may need to click View more
Debugging the server side

Prerequisites

Steps to debug the server side

1. On the web client exposed by the Azure-hosted development machine, ensure that there are mobile workspaces published for the Finance and Operations app. For information about publishing a mobile workspace, see Publish mobile workspaces.

2. Open the app on your device, point to the Azure-hosted development machine, and sign in.

3. Open Visual Studio on the Azure-hosted development machine and attach the debugger to the w3wp process.

4. After you connect the debugger, find the desired business logic, and insert breakpoints as needed.

5. Either use the app on your device as usual, or reflect the mobile device on your desktop so you can interact with it on the desktop screen.

6. Navigate through the desired workspace and forms.

5. Reflect the mobile device on your desktop so that you can interact with it on the desktop screen.

6. Go through the desired workspace and forms.

7. If breakpoints are encountered, then the browser developer tools will allow you to control the flow of execution and see the values and parameters being passed.

8. To change styling at runtime, use the elements tab to alter the styling. This will help you determine what elements JavaScript should target and how those elements should be styled.

9. If a needed change is identified, make those changes in JavaScript, and then push those changes into the environment.

10. If more changes or validation is needed, repeat the process.

**Debugging the server side**

**Prerequisites**

- Azure-hosted development machine (so the mobile device can point to it)

**Steps to debug the server side**

1. On the web client exposed by the Azure-hosted development machine, ensure that there are mobile workspaces published for the Finance and Operations app. For information about publishing a mobile workspace, see Publish mobile workspaces.

2. Open the app on your device, point to the Azure-hosted development machine, and sign in.

3. Open Visual Studio on the Azure-hosted development machine and attach the debugger to the w3wp process.

4. After you connect the debugger, find the desired business logic, and insert breakpoints as needed.

5. Either use the app on your device as usual, or reflect the mobile device on your desktop so you can interact with it on the desktop screen.

6. Navigate through the desired workspace and forms.
7. If breakpoints are encountered, then Visual Studio will allow you to control the flow of execution and see the values and parameters being passed.

8. If a needed change is identified, make those changes in X++, and push those changes into the environment.

9. If more changes or validation is needed, repeat the process.

Troubleshooting the app

[Resolved] - No support for iOS14 due to issues with date and time controls

Version 2.2.8 of the Finance and Operations mobile app fixes the known issues with the date and time pickers in iOS14. Ensure that you have the latest version of the app if you are experiencing issues running the application on iOS14.

The Mobile Client app is not working on particular devices

Sometimes the cache associated with the app becomes corrupt or obsolete and needs to be cleared. Unfortunately, the only way to clear the data associated with the app is to uninstall the app. To completely uninstall the app, don’t use the “long-press wiggle and x on the app icon” method. Instead, completely uninstall the app by navigating to Settings > General > iPhone Storage > Finance and Operations (Dynamics 365), and then click Delete App. After 10-15 seconds, the app can be reinstalled.

On Android devices with non-English regions, the comma can’t be used as the decimal separator in an amount field

On Android devices with non-English regions, using a comma as the decimal separator is standard practice. Problems using a comma in an amount field is an Android-specific problem because iPhone works as expected. On Android, use of the comma in an amount field is a problem with the default “gboard” keyboard and some other keyboards. Installing the SwiftKey keyboard (published by Microsoft) allows the entry of commas just like on iPhone: SwiftKey Keyboard.

Change needed for ADFS to support Mobile Client in on-premises environments

If Active Directory Federation Services (ADFS) is in use on the domain and the environment is on-premises, then ADFS must be configured to provide a regular forms-based authentication screen instead of using Windows Integrated Authentication (WIA). The Finance and Operations apps for iOS and Android require the regular forms-based authentication screen. ADFS should be configured to only provide WIA for browser clients (use cases). For more information, see Configure intranet forms based authentication for devices that do not support WIA.

Using multi-factor authentication with the Finance and Operations app

The Finance and Operations (Mobile Client) app facilitates user authentication with Azure Active Directory (Azure AD) by presenting the Azure AD sign-in web page within an embedded browser. After a successful sign in, it will retrieve the user token from the cookies and use that when communicating with the user interaction service that it shares with the web client. Some multi-factor authentication mechanisms that involve switching to a different app on the same device will cause the embedded browser to close, so the sign in will fail. The workarounds for this include:

- Different device - Use a different device for the multi-factor authentication response so the app remains active on the original device.
- Multi-factor authentication via phone call - Use a phone call for the multi-factor authentication response so an app switch is not needed.
- Use the “touch and hold” gesture on the authentication notification and then select the Accept option. Because the notification acceptance will not require an app switch, the sign in will proceed as usual.

If there are continued problems with MFA authentication, it is helpful to submit the Microsoft Authenticator app
logs and provide support with the resulting Incident ID.

Intune support and conditional access

The Finance and Operations (Mobile Client) app does not have Microsoft Intune policies implemented, so it does not support Intune. Manually adding the app (following Add iOS store apps to Microsoft Intune) is also not supported because the device identifier cannot be passed.

Trouble signing out of the app and signing in with new credentials

If you experience trouble signing out of the app and signing in with new credentials, then you might need to "forget old credentials" on the Azure AD sign-in screen.

- To sign out of the app, follow these steps:
  - Open the app.
  - Sign out of the app.
  - Force close the app.
- To forget old credentials, follow these steps:
  - Open the app.
  - Connect to the server.
  - On the Azure AD sign-in screen, if there are saved credentials, select the ellipsis (…) button on that card, and then select Forget the credential.
  - Force close the app.
- To sign in to the app, follow these steps:
  - Open the app.
  - Connect to the server
  - Sign in using the Azure AD sign-in screen.

Troubleshooting app content

I can't figure out how to build or change something in my Mobile Client content

There are many resources that you can leverage to figure out how to build or change content for the Mobile Client.

- Review the documentation provided in the Help system.
- Review the Fleet Management Samples for examples.
- Publish and review the Expense Management workspace, and other standard workspaces, for examples. Demo data for the USSI company is useful when using the Expense Management workspace. The forms and X++ code that make up the Expense Management workspace can be found in the Application Explorer by searching for the "ExpenseMobile" prefix.
- Leverage the Dynamics Community forums by searching for answers and asking questions when needed.

Tips for workspace creation and modification

Here are some tips for workspace creation and modification:

- Create new simplified forms for recording rather than recording large complex forms.
- After recording a form, you have to close the form instead of clicking Done, otherwise the form remains open.
- Verify that recordings are correct using the "Job steps".
- Play back recordings using task recorder playback to verify them.
- Don’t navigate to a page before starting the recording, because the context from the previous page might need to be captured.
- If you re-record a page with a grid then you need to re-record the link to the Details page because otherwise it won't be there.
- When recording an action, change the value of the fields to add them. When recording is complete, close the form instead of clicking **Save**.
- Lookups in mobile are list pages that have been recorded. **Select field data** and **Select field to display** are used to select the **field to use as the value to save** (data) and the **field to show the user** (display).
- When adding a lookup field, select a value in a lookup instead of just adding the lookup field. This will ensure that the correct value is selected.
- If you re-record a lookup, all the references also need to be re-recorded because the GUID for the lookup will change.
- If you want to add a field to a page, you need to add all the fields again, because the list is cleared at the beginning of each edit. This is a limitation of task recorder. Note that reordering is also not possible.
- In the workspace XML, GUIDs are used as references to forms and controls instead of names. GUIDs are used to ensure uniqueness, but this comes at the cost of maintainability. Those GUIDs are regenerated on each modification, so partial edits are very difficult. The use of GUIDs would be very costly to change, so it is unlikely that changes would be made in the future to use simpler string name references.
- Relationships between form datasources need to be via RecordId instead of string. For example, primary keys of the datasources should not be strings.
- Customers and partners can fork a workspace by creating a copy of it and then make changes as needed.
- There is no check box in mobile. You have to manually bind the field to a Yes/No enum in JavaScript.

**Common problems with form recordings**

Avoid using forms with these patterns and controls when creating workspace recordings:

- Datasources with DelayedJoin (common on transaction forms).
- FastTabs (common on existing forms).
  - Recorded forms should not have FastTabs because the FastTabs expansion state can interfere with playback.
- Any user interface (UI) that has state, like an expandable or hide/show region.
- There is no check box in mobile. You have to manually bind the field to a Yes/No enum in JavaScript.
After you acquire a development environment, complete the following procedures to get started with development.

**Get the Fleet Management mobile forms**

We have created new, purpose-built forms in the Fleet Management module. These forms are used specifically for the mobile app and aren’t meant to be used through the web client.

1. Download the file that contains the Fleet Management project (.axpp file).
2. Extract the contents of the zip file to a temporary location on the development computer.
3. Import the project (.axpp) file by using Microsoft Visual Studio (click Finance and Operations > Import Project).
4. After you've imported the project file, build the project or module.

**Get the sample workspace**

We provide a sample workspace for Reservation management. This workspace is based on the Fleet Management module.

1. Download the file that contains the sample workspace (.xml file).
2. Sign in to your non-production client. (You must sign in as an administrator.)
3. In the address bar, add &mode=mobile to the end of the URL, and then press Enter.
4. In the client, go to Settings > Mobile app. The mobile app designer will appear docked next to the client.
5. Click the Overflow button (…), and then click Import.
6. Click the Browse button that appears at the bottom of the page.
7. In the file selection dialog box that appears, select one of the XML files that you previously extracted from the zip file.
8. After the app has been loaded into the mobile app designer, click Done at the bottom of the page.
9. Click Publish workspace.

**Get the mobile app**

The mobile app is being made available for the most popular mobile operating systems. You must have a Dynamics 365 Unified Operations instance and valid user credentials in order to log in to the app.

- Android (available now) - Finance and Operations mobile app on the Google Play Store
- iPhone (available now) - Finance and Operations mobile app on the iTunes apps store

You’re done! Launch the app from your mobile device to see the sample workspace.

**Additional resources**

*Architecture*

*Client APIs reference*

*Server APIs reference*
The mobile app communicates with Application Object Server (AOS) to get the metadata for the mobile workspaces (and the pages and the fields that appear on the page), and to get the data for the fields on the pages. Each time that the mobile app requests data for a page, AOS creates a new session that uses the context of the user who is using the mobile app. AOS then uses the user’s context to open the corresponding forms (by using the corresponding menu items). AOS can open multiple forms in quick succession and perform actions on those forms (for example, filtering, opening FactBoxes, changing tab pages, and clicking buttons). Any business logic on the forms is also run as usual. Through that process, AOS collects the data values from the requested fields and then sends that data back to the mobile app.

The mobile app platform doesn’t assume connectivity to Finance and Operations apps. Activities such as navigation, data view, and data entry don’t require server connectivity after data has been cached.

Understanding navigation in the mobile app

Navigation in the mobile app consists of four simple concepts: the dashboard, workspaces, pages, and actions.
When you start the app, you land on the **dashboard**. On the **dashboard**, you can see a list of **workspaces** that are published in your environment.

In each **workspace**, you can see a list of **pages** that are available for that workspace.

On a **page**, you can view data that is collected from one or more forms.

From a **page**, you can navigate to other **pages** for related data, such as an entity details or lines.

On a **page**, you can see a list of **actions** that are available for that page.

**Actions** let you create or edit existing data.

**Notes**

At any time, you can pull-to-refresh in the mobile app to make the mobile app update its data or metadata. After you edit an existing workspace or publish a workspace, be sure to pull-to-refresh in the mobile app, in either the list of workspaces (if you added a workspace or business logic) or the list of pages (if you modified a page or an action). Workspaces that have been published are visible to all users. In Platform update 3, menu item security automatically hides pages that the user doesn’t have access to. If a user doesn’t have access to any pages in a workspace, the workspace itself is hidden.

**Using the mobile app designer**

The mobile app designer lets you select the specific data fields from forms that should appear in the mobile app.

A mobile workspace can be created through designer, using X++ attribute APIs or a combination of both. See [Configure workspaces by using the SysAppWorkspace class](#) for more details on using X++ APIs for building a mobile workspace.
1. Open the client.
2. Go to Settings > Mobile app.
3. Create a new workspace, or select an existing workspace to edit.
4. Specify the name of the workspace, an icon, and a color.
5. Add pages to the workspace, or edit an existing page.
6. Specify the name of the page.
7. Click Select Fields to select the data fields to add to the page.
8. Open the forms that have the data fields that you want to add, and then click the yellow plus sign (+) that appears next to the fields. The fields are added in the order that you select them in. You can add fields from multiple forms, in any order.
9. When you’ve finished selecting fields, click Done.
10. If you’ve added a field list to the page, you will see that the List type is specified for one of the items in the field list. You can optionally add a details page for items in that list by following these steps:
   a. Select the list by clicking on it in the designer.
   b. Click Add details page.
   c. Repeat steps 6 through 10 as you require.

**Refresh the app after you make changes**

<table>
<thead>
<tr>
<th>TYPE OF CHANGE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>TYPE OF CHANGE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>New workspaces, deleted workspaces, or changes to the name, color, or icon of a workspace</td>
<td>Pull-to-refresh from the main landing page (dashboard) of the app, where you see the list of workspaces.</td>
</tr>
</tbody>
</table>

- **Sales order management**: Manage your sales orders and customers
- **Reservation management**: Manage your vehicle fleet and reservations
- **Supplier Management**: Manage your supply chain
- **Approvals & Workflows**: Manage your invoices and workflows
- **Warehousing & Logistics**: Manage inventory and shipping logistics
- **Procurement & Sourcing**: Manage purchases and suppliers
- **Sample sales order**:
<table>
<thead>
<tr>
<th>Type of Change</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All other changes (new or changed pages or actions, or changes to business logic)</td>
<td>Pull-to-refresh from the workspace that has the edited pages or actions.</td>
</tr>
</tbody>
</table>

### Additional resources
- Page design guidelines
- Action design guidelines
- Form design requirements
Mobile workspace business logic events provide places for developers to specify workspace configuration to enhance capability, and implement business-scenario-specific behaviors. All mobile business logic executes in the process of the mobile app, and business logic execution flow is controlled by the Operations mobile app framework.

Code that executes in business logic can make runtime modifications to the metadata of Pages, Actions, and Controls. These runtime modifications overlayer the static metadata that is cached in the app, but the modifications do not directly change the cached static metadata, nor do they affect the static metadata that is stored on the server. These runtime modifications only persist until the app is closed.

Code that executes in the business logic of the app can make runtime modifications to business data that is stored in the app. These modifications (when performed according to correct practices) participate in the normal data processing framework as other business data in the app. That means adhering to the offline-first capabilities and asynchronous save behaviors provided by the framework.
Before you begin to use the designer to build pages and actions, it's important that you plan the overall design of the mobile workspace that you want to build. We recommend that you orient your design around the entities that you plan to use in the mobile workspace. Don't begin by thinking about the forms that you want to use. From the perspective of the mobile app, the forms are just a mechanism for retrieving data, and the run-time UI behavior of a form isn't applicable to the mobile app. Therefore, you should first identify your entities and the relationships between them. For each entity, the following questions will help you decide how you should design your forms and pages.

**How do I create a list view for an entity in the mobile app?**

1. Identify or create a form in the web client that contains a grid for the entity.
2. Make sure that the grid is bound to the table that represents the entity.
3. Make sure that the form has a menu item that is root-navigable.
4. Make sure that the form can be opened directly via a URL that includes the menu item parameter.
5. Make sure that the filter pane enables the grid to be filtered based on the desired fields.
6. In the designer, create a page for the entity.
7. In the designer, put only a list on the page.
8. In the designer, put the desired fields in the list on the page.

**What if I don't want a list view for this entity?**

If you want just a details view for an entity, it's likely that the entity is a singleton for a given context (such as a given user or a given company). This pattern applies, for example, to a details view for an employee's own profile in a self-service workspace or a details view for the company context that is used for the current session. See the guidelines for creating a page for a details view.

**How do I create a details view for an entity in the mobile app?**

1. Identify or create a form in the web client that contains the details view for this entity.
2. Make sure that the Master Root Data Source on the form is bound to the table that represents the entity.
3. Make sure that the form has a menu item that is root-navigable.
4. Make sure that the form can be opened directly via a URL that includes the menu item parameter.
5. In the designer, create a page for the entity.
6. In the designer, put the desired fields on the page.

**How do I create list-to-details navigation for an entity in the mobile app?**

1. Make sure that you've created both a list view page and a details view page for the entity by using the designer.
2. Make sure that the entity for the list view is the same as the entity for the details view. In other words, the table that is bound to the grid on the form that is used for the list view must be the same table that is the Master Root Data Source on the form that is used for the details view.
3. Make sure that the form that is used for the details view can be filtered on a unique key field by using the filter pane.
4. In the designer, make sure that the list view page is linked to the details view page. Click the list, open the properties, and then set the details view page by using the lookup.
How do I add a reference field that enables navigation to a related entity?

1. Make sure that either a list view page or a details view page exists for the entity that contains the reference.

2. Make sure that the page contains the reference field from the entity that is being referenced.

3. Make sure that the referenced field is bound to the referenced entity’s data source, and that the referenced entity is *outer joined* (1-0..1) or *inner joined* (1-1) to the data source for the entity that contains the reference. For example, in the following illustration, FMRental is the entity that contains the reference, and FMVehicle is the referenced entity.

4. Make sure that you've created a separate details view page for the entity that is being referenced.

5. Make sure that the reference field has been added to the page.

6. In the designer, make sure that the reference field has been linked to the details view for the referenced entity. For example, in the following illustration, Vehicle-details is the details view page for the referenced entity.
How do I add a list that contains items from a related entity to a details view page?

1. Identify or create a form in the web client that contains the details view for the entity, and make sure that the form adheres to the guidelines in the “How do I create a details view for an entity in the mobile app” section.

2. Make sure that the form contains a grid that is bound to the table that represents the related entity.

3. Make sure that the table for the related entity is active joined to the table for the entity that contains the reference.

4. Create a details view page that contains the desired fields for the entity, and that also contains a list that has the desired fields from the related entity.

How do I make the list show up in-line in the details view?

1. Identify or create a form in the web client that contains the details view for the entity, and make sure that the form adheres to the guidelines in the “How do I create a details view for an entity in the mobile app” section.

2. Make sure that the form contains a grid that is bound to the table that represents the related entity.

3. Make sure that the table for the related entity is active joined to the table for the entity that contains the reference.

4. Use the form to create a details view page that contains the desired fields for the entity.

5. Use the same form to create a separate list view page that contains only a list that has the desired fields from the related entity.

6. On the details view page, add a PageLinkControl that links to the list view page. Currently, you must use business logic to add the PageLinkControl. The following example show the code that Fleet Management uses.
1. Identify or create a page that contains the controls with the data that you want.

2. Refer to the following code example, which hides the page from the navigation menus, and accesses data on the page using the provided APIs. Note that 'My-Hidden-Page' and 'My-Field-Id' are the names of the page and control, respectively, and can be found when viewing the corresponding page in the designer.

```javascript
function main(metadataService, dataService, cacheService, $q) {
    myField1Value = ''; // This variable will be populated in appInit, and can then be used elsewhere in the business logic.
    return {
        appInit: function (appMetadata) {
            var myHiddenPage = metadataService.findPage('My-Hidden-Page');
            if(myHiddenPage) {
                var dataPromise = dataService.getPageData(myHiddenPage.Id,'','',0);
                dataPromise.then(function (result) {
                    var myField1Id = metadataService.findControl(myHiddenPage, 'My-Field-1').Id;
                    myField1Value = result.data[myField1Id];
                });
            }
        }
    }
}
```

**How do I adjust the number of records returned in a list page using list fetch size?**

The number of records returned in a list page is controlled by the List fetch size value. The default is 50 records. The List fetch size indicates the maximum number of records returned by a page when it first loads, and the maximum number of records returned when search is used to find a specific set of records. Be careful not to make the value too large or it may negatively affect the user experience.

1. In the Mobile App designer, add a page containing a grid and select some fields from the grid.
2. Click the Grid node and then click Properties.
3. The Control properties dialog box will contain a default fetch size of 50 records.
4. Adjust the fetch size as needed.
Actions let users create, update, or delete data, and also run business processes on that data (such as submit, confirm, and post). A user who completes an action first supplies the data for the action (if the action accepts data input). When the user has finished supplying the data, the action is put into a queue of similar actions (which are sometimes referred to as data sync operations). If the device is connected/online, the queue is processed immediately. Otherwise, it’s processed the next time that the device is connected. The queue is processed asynchronously and doesn’t require the user’s attention unless there is an error during data synchronization. Errors of this type can occur because of server-side data validation. Actions are powered by a server-side mechanism that resembles Task recordings. This mechanism extracts the user’s input from the action and then automatically runs the business process steps on the server by using the input values that the user supplied. The mechanism automatically opens forms, clicks buttons on the forms, and enters the user’s input into controls on the forms. This process of playing back the action against the forms on the server occurs asynchronously, against “headless” forms. The mobile app informs the user when the process is completed, and shows the user any info, warning, or error messages that the forms logged. When you design an action, it’s important that you first consider what entity the action is related to. In the current framework, an action must operate on only one entity. An action should not update multiple entities at the same time. For example, an action to create a new sales order should create only the header for the order. It should not also try to create lines, because the lines are separate entities. When you decide to design the action, consider the following questions to determine how to proceed.

**How do I design an action that enables an entity to be created?**

1. Identify or create a list view page for the entity.
2. Make sure that the form that is used for the list view page includes a **New** button that can be used to add new records to the list.
3. Use the designer to create a new action for the page. While you’re designing the action, be careful not to perform any unnecessary actions. Enter data only in those fields that should be available to the user, and click only those buttons that are required (for example the **New** button and the **Save** button).

**How do I design an action that enables an entity to be edited?**

1. Identify or create a details view page for the entity.
2. Make sure that the form that is used for the details view page includes an **Edit** button that can be used to edit the visible record.
3. Make sure that the form that is used for the details view page lets users open a specific record by applying filters in the filter pane.
4. Use the designer to create a new action for the page. While you’re designing the action, be careful not to perform any unnecessary actions. Enter data only in those fields that should be available to the user, and click only those buttons that are required (for example, the **Edit** button and the **Save** button).

**How do I design an action that enables an entity to be deleted?**

1. Identify or create a details view page for the entity.
2. Make sure that the form that is used for the details view page includes a **Delete** button that can be used to delete the visible record.
3. Make sure that the form that is used for the details view page lets users open a specific record by applying filters in the filter pane.
4. Use the designer to create a new action for the page, and just click **Delete** as a part of the process of designing the action.
How do I design an action that enables a business action to be performed on an entity?

1. Identify or create a details view page for the entity.
2. Make sure that the form that is used for the details view page includes a **Delete** button that can be used to delete the visible record.
3. Make sure that the form that is used for the details view page lets users open a specific record by applying filters in the filter pane.
4. Use the designer to create a new action for the page, and just perform the steps that are required in the business action. You don’t necessarily have to enter data in fields as part of the action. For an action such as *submit*, you just have to click the **Submit** button (and acknowledge any confirmations that appear).

How do I design an action that enables a field value to be set via a rich lookup?

Lookups for fields in the mobile app don’t have a correlation to the advanced lookup behaviors in the cloud version of the app. Regardless of whether you have a custom lookup in the cloud version of the app or an automatic lookup that uses a simple query, the mobile app doesn’t run existing lookup code when it must determine which UI to show the user. (Remember that the user might be offline while using the app, and server-side code isn’t run until the action is synchronized.) However, lookup/control overrides such as *modified* are run when the value is set by the mobile back end as it synchronizes the data from the action. When the mobile app detects that a field on an action was selected from a lookup field on a form, it shows a device-native combo box/list picker control, and populates the items by directly querying the backing table of that lookup field. The items in the list show the user data from the TitleField for records in that table. Follow these steps to add the rich lookup experience to your action. This lookup experience includes a full-page multi-column lookup selector that has offline search.

1. Identify or create a list view page for the entity behind the lookup. You can reuse existing list view pages that you’ve already created.
2. After you’ve finished designing the action, select the field to add rich lookup functionality to, and then click **Properties**.
3. In the **Control properties** dialog box, select the list view page that you identified or created in step 1, and set the other related properties.
Manage mobile app

Action title
New Reservation

Action description
Create a new reservation

Action name
Add-Reservation

BUILD AN ACTION

+ Select fields

Control properties

Control name
FMRental_Customer

Control label
CustomerSS

ADD LOOKUP PAGE

Select existing page
All-Customers

Select field to use
Record-ID

Select field to display
Customer

Done
4. Save and publish your changes to the action.

If you don’t see the property for the existing list view page or can’t access the Control properties dialog box when you’re designing your action, you might be using an older build of the app. In this case, you can still add rich lookup functionality by using a business logic file.

```
function main(metadataService, dataService, cacheService, $q) {
    return {
        appInit: function (appMetadata) {
            metadataService.configureLookup(
                // specify the name of the Action to add the lookup to
                'Add-Reservation',
                // specify the name of the Action's field to add the lookup to
                'FMRental_Customer',
                {
                    // specify the name of the Page for the Entity for the lookup
                    lookupPage: 'All-Customers',
                    // specify the Page's field which contains the value to set on the lookup
                    // this value should be the same value you can type into the field on the Form
                    valueField: 'FMCustomer_RecId',
                    // specify the Page's field which contains the value to display to the user
                    // this value is only used for display. The value field is passed to the Form
                    displayField: 'FMCustomer_FullName',
                    // set this to true to enable the rich lookup
                    showLookupPage: true
                }
            );
        },
    );
}
```

How do I prevent an action from appearing in the list of actions for a page?

To prevent an action from appearing in the list of actions for any page, call the following code from the `appInit()` section of your business logic. In this code, `action-name` is the name of your action (as specified in the Action name field in the designer).

```
function main(metadataService, dataService, cacheService, $q) {
    return {
        appInit: function (appMetadata) {
            metadataService.configureLookup('action-name', { visible: false });
        },
    );
}
```
This section provides valuable guidelines for building forms that work well with the mobile app.

- Each form must have an associated Display Menu Item.
  - The Display Menu Item's **Allow Root Navigation** property must be set to **Yes**. This setting enables the mobile framework to open the form that is referenced by the menu item.
- Each form must be directly accessible via its Display Menu Item.
  - To verify accessibility, open the menu item via a URL. Just append `&mi=` to the URL, where the value is the Application Object Tree (AOT) name of your menu item.
  - If the form doesn't open or show data when you access it in this way, the form won't work with the mobile app.
- Each form that shows data must have one Master Root Data Source.
  - This data source must be the first data source on the form (top-most in the designer).
  - This data source must not be joined to any other data sources.
- Each form must work with the data source filters.
  - After you open the form in the web client, open the filter pane by using the **Show filters** button.

Then click **Add a filter field**, and verify that the Master Root Data Source appears as the table for fields in the list of available fields. Other tables can also appear, but the Master Root Data Source **must** appear in this list. Otherwise, the mobile app won’t enable searches and navigation that uses context.

- Searching: The mobile app does online searches against data by using the Filters framework behind the scenes.
- Navigation that uses context: The mobile app enables list-to-details navigation (and other context-aware navigation) by first opening the target form via the menu item and then using the Filters framework to show only the specified record context.
- List-to-details navigation: The table that the grid is bound to on form A (the list form) must be the Master Root Data Source on the details form (form B). When a user selects a record in the list on form A, the mobile framework navigates with record context by applying filters on form B that uniquely identify the record.

**Design considerations**

**Using data methods**

You can use display methods to show data on pages (both list type pages and detail type pages). However, there
are two key points to remember when you use display methods:

- **Searching** – When a user performs an "online" search (that is, a search that is run against data in the web client instead of locally cached data), the search won’t match against display methods, because the Filtering framework in the web client doesn’t support searches against data methods. However, when a user does a search against locally cached data, the search will match against display methods, provided that the records have been cached on the device.

- **Offline** – If a user creates or updates data while their device isn’t connected to the server, temporary records are created in the local cache. Because these temporary records haven’t yet been processed in the web client, if the records have any fields that are automatically populated or defaults by server-side business logic, these fields will remain empty until the records have been synced with the web client. Display methods fall into this category of fields that will be empty for a temporary record.

**Designing for offline**

Unlike the web client, which is highly connected to the server and maintains an open user session that has open forms on the server, the mobile app creates user sessions (and opens forms) only in short bursts while the app is being synced with the server (via data read for pages, or via data write/update for actions). If there are no actions to sync with the server, and if the local data cache is up to date, the mobile app won’t communicate with the server as a user navigates around the app (unless the user triggers an explicit pull-to-refresh). It’s important that you keep this data flow pattern in mind while you design pages and actions in the mobile app. You should not expect form logic to run every time that a page is loaded or an action is started. You should also never expect form logic to run while a user is completing an action. Form logic is run only when the action is being synced with the server. The following list describes the only times when you should expect Form logic to run.

**Form logic runs right before a page is opened on the mobile app for the first time.**

1. When a user first opens a page, the mobile app reaches out to the web client and opens the associated forms. *During this process, logic such as form init and data source init is all run in the usual manner.*
2. The mobile app framework reads the required data directly from the controls on the forms and sends the data back to the mobile app.
3. The mobile app caches the data and shows it in the page on the mobile app.
4. Future attempts to open the page will load the cached data. *These attempts won’t run the form logic again, unless the user explicitly refreshes the page or the cache expires. (Currently, the cache set to expire after 30 minutes.)*

**Processing an action that has been submitted to the server from the mobile app**

1. When a user opens an action and fills in the data in that action, *no form logic is run.* A user can complete an action either offline or online. The system behaves the same way in both cases.
2. After the user clicks **Done/Save** on the action, the mobile app queues a data synchronization operation. This operation will be synced with the server when the mobile app is connected to the Internet.
3. When an Internet connection is detected (which can happen immediately after the action is completed) the mobile app sends the data synchronization operation to the server for processing.
4. While the operation is processed on the server, the framework opens the associated forms and enters the data from the action by passing values into the form controls. *During this process, form logic is run in the usual manner (init, modified, clicked, and so on, are all run).* However, the mobile user might have moved to a different part of the app while this processing is occurring. *Any form logic that shows/hides controls will have no effect on the UI that is seen in the mobile app.* Therefore, to minimize synchronization times, it’s best not to include any UI logic on the form.

**Building mobile versions of existing forms**

If you decide to modify existing forms so that they work with the mobile framework, instead of building new mobile-specific forms, you might have to conditionally change the form’s behavior for mobile-specific scenarios. You can use the following static X++ application programming interfaces (APIs) in your X++ code to determine whether the code is being accessed during a session where a web client user is designing pages/actions or...
during a session that the mobile framework back end created to load pages/actions for a mobile user. **When a form is being used with the mobile designer**

SysTaskRecorderController::isDesigningApp()

**When a form is being used by the mobile framework back end to load pages and run actions**

SysTaskRecorderController::isExecutingApp()

**Form control support**

The form controls for the various base data types (strings, dates, and numbers) and grids are supported. However, a few common controls have limited support. **Reference groups** Fields from within Reference groups controls are compatible when you design pages. However, they aren't compatible when you design Actions. Although you might be able to select these fields without experiencing any issues, Reference groups have a fundamental incompatibility with the mobile framework. We recommend that you not use Reference groups. Instead, add a control directly to the form, and then bind the control directly to the surrogate foreign key (SFK) by using the property sheet.
Configure workspaces by using the SysAppWorkspace class

Workspace class, **SysAppWorkspace**, is the starting point to create, configure and publish workspaces on the server. The following categories of APIs are available for use in sysAppWorkspace:

- **Workspace attributes** - This is used to create pages, tasks, entities, lookups, relationships in order to build mobile workspaces.
  
  - Download the sample project for Fleet Management Mobile App. This is an .axpp file found at [Dynamics365-for-Operations-mobile-FleetManagementSamples](Dynamics365-for-Operations-mobile-FleetManagementSamples).
  
  - After downloading the file, open Visual Studio on your Operations development environment, select *Dynamics 365 > Import Project*, and browse for the downloaded project file. On the same dialog box, select *Overwrite* and select *Create a new solution*. After the import is complete, build the solution (or build the Fleet Management model).
  
  - To review the example, start by reviewing the FMReservationManagementWorkspace class to see all the pages and actions included in the workspace. Use Solution Explorer to find page and task classes, and all the assets included in each. Use the API reference for more details on each API.
  
  - A mobile workspace can be created through designer pane, using X++ attribute APIs or a combination of both. See the "Use the workspace class to publish workspaces from AOT resources" section below for more details about how to import mobile app metadata from designer to AOT. The sample project Fleet Management Mobile App is a complete mobile app built using X++ attribute APIs.

- **Workspace metadata classes** - This is used to inspect and apply server-side business logic to metadata for mobile workspaces.

For a complete list of server-side APIs, see [Server-side development (workspace X++ APIs)](Server-side-development-workspace-X--APIs).

Create a new workspace class

To use the **SysAppWorkspace** class for your workspace, you must create a new class for the workspace by extending the **SysAppWorkspace** class. You can then use the new class to modify workspace metadata. The new class also provides hooks for life cycle management of the mobile app.

Follow these steps to create a new workspace class for your workspace.

1. Create a new class for your workspace, and extend it from the **SysAppWorkspace** class.

   ![MyNewWorkspace class extending SysAppWorkspace](MyNewWorkspace.png)

2. Add the **SysAppWorkspaceAttribute** attribute to the new class, and provide the **AppID** value of your workspace. You can find the app ID for your app on the *Summary* page in the mobile app designer.
3. Optional: If your workspace is an Application Object Tree (AOT) resource, provide the AOT resource name as the second parameter for the `SysAppWorkspaceAttribute` constructor.

Use the workspace class to publish workspaces from AOT resources

Workspaces can reside in the database. They can also reside in the AOT as resources. To provide visibility into workspaces that are stored in AOT resources, you must create a workspace class and point it to the name of the AOT resource that contains the workspace. Workspaces that are stored as AOT resources can't be edited or deleted by using the mobile app designer. Those workspace can only be exported.

Follow these steps to publish a workspace that resides in an AOT resource.
1. When you're developing a workspace that is stored in the database, you must export it from the mobile app designer so that it can be stored as an AOT resource. The workspace is exported as an XML file.

2. Delete the workspace from the mobile app designer. You will load it from an AOT resource later.

3. Create a new AOT resource, and select the exported workspace for the resource.

4. Create a new class for your workspace. This class should extend `SysAppWorkspace`. Apply the `SysAppWorkspaceAttribute` attribute to the class, and provide the app ID and the AOT resource name that contains the resource.

5. Build the class, and reopen the mobile app designer.

   The workspace is now published. It appears in the designer, but can't be edited or deleted. Note that the workspace is loaded from metadata.
Update a workspace that has already been published

If your workspace is part of an AOT resource, you can't edit it by using the mobile app designer. In the following example, a workspace that is named **MyWorkspace** exists in the AOT, and it has a backing class that is named **WorkspaceInAOT**.

Follow these steps to edit the workspace.

1. Export the workspace by using the mobile app designer. The designer automatically creates new app IDs for workspaces that are stored in the AOT.

2. Import the newly exported workspace by using the mobile app designer.
   
   a. Optional: Change the name so that the newly added workspace can be distinguished from other workspaces.
   
   b. Copy the app ID of the newly created workspace.
3. Create a new class that extends your backing class, apply the `SysAppWorkspaceAttribute` attribute, and specify the new app ID.

You can now continue to work with your new workspace and the backing class. After you've finished making your changes, you can merge them with the AOT-based workspace.

### Delete a workspace that is an AOT resource

When a mobile workspace is stored as an AOT resource, you can’t delete it by using the mobile app designer. Follow these steps to delete a workspace that exists as an AOT resource.

1. Delete the AOT resource that contains the workspace.

2. Delete the workspace class that was created for the workspace.
3. Do a full model build that contains the AOT resource and the class. The following illustrations shows a full build of the Application Foundation model. The Application Foundation model also contains the AOT resource and workspace class. To speed up the full build, you can clear all the selections on the **Options** tab.

4. When the build is completed, reopen the mobile app designer, and verify that the workspace is no longer there.
The fields on a mobile app page can be customized so that they are shown as email addresses, phone numbers, or URLs.

**Email field**

You can mark a field as an email address field by using business logic. Then, when a user clicks the field, the default mobile email app starts, and the field value appears as the email address in the app.

**Phone field**

You can mark a field as a phone number field by using business logic. Then, when a user clicks the field, the mobile dialer app starts, and the field value appears as the phone number in the app.

**NOTE**

On iOS, if the phone number isn't valid, it might not trigger the mobile dialer app.

**URL field**

You can mark a field as a URL field by using business logic. Then, when a user clicks the field, the URL opens in the default mobile browser, and the field value appears in the address bar.

**NOTE**

On iOS, you must provide a complete URL (that is, a URL that starts with a protocol, such as https). Otherwise, the URL isn't opened in the browser. A URL such as www.microsoft.com doesn't work. Instead, the URL must be specified as https://www.microsoft.com.

**Example**

This example shows how to configure the customer email address and phone number fields so that they can clicked and opened in the appropriate iOS apps.

Before the fields are customized, they can't be clicked, as shown in the following image.
Follow these steps to specify that a field is a link.

1. Add the following lines to the `appInit` method. You call the `configureControl` method, and pass in the page name and control name. You then supply the `LinkType` value for the control. The following values are supported: `Telephone`, `Email`, and `Url`.

   ```javascript
   metadataService.configureControl('PageName', 'ControlName', { LinkType: 'Telephone' });
   metadataService.configureControl('PageName', 'ControlName', { LinkType: 'Email' });
   metadataService.configureControl('PageName', 'ControlName', { LinkType: 'Url' });
   ```

2. Upload the updated business logic file by using the mobile app designer.

3. Update the workspace metadata in the mobile client.

The fields now appear as links.
Terrance Betz

Driver's license

E-mail
someone@example.com

Phone

Actions
Although a pageLink control can be used to show counts (totals), it can be slow, because it must load the target page before it counts the number of rows. Additionally, the count that is calculated can be incorrect, because there is a limit on the number of rows that are retrieved.

If you want to make mobile workspaces work more quickly, we recommended that you use a regular field to show the count and then model the field as a pageLink control in the mobile client.

The following example uses the Fleet Management app. In the Fleet Management app, the workspace shows the total number of customers, reservations, and vehicles. Previously, these counts came from a pageLink control that had AllCustomers, AllReservations, and AllVehicles as targets. The pageLink control loaded the rows and did the count. (This approach isn't the recommended approach.)

Follow these steps to configure the workspace page to use the recommended approach.

1. On the server, create a new form to contain fields that are also on the server. (You can also add the new fields to an existing form). In the following illustration, a new FMMobSummary form is created that has three fields.
2. Create a page by using the mobile app designer for the FMMobSummary form.

3. Update the business logic to transform the fields into a pageLink control. Use the configureControl method to add a navigation target to the fields. The fields are then configured as pageLink controls. The arguments for the configureControl method are the page name, the control name, and an object of properties that must be updated.

4. Update the workspace design. Embed the summary page as a part in the workspace page. Reference the fields that are now configured as pageLink controls. Provide a style, and set the showCount:true property, so that the count is shown on the pageLink control.
By using this approach, you also get the localized labels for `pageLink` controls. The result is a much faster experience when workspaces are loaded.
This topic describes how to limit a user’s access to a workspace.

**Assign a menu item to workspace**

Workspaces can be tied to a menu item. Users who don’t have access to the menu item can’t use the workspace, because the workspace is shown only to users who have rights to the menu item.

If a menu item isn’t assigned to a workspace, the workspace is always shown to the user.

Follow these steps to help secure your workspaces by assigning a menu item.

1. Add a `SysAppWorkspaceSecurityAttribute` attribute to the workspace class, and specify the menu item to assign to the workspace.

   ```java
   [SysAppWorkspaceSecurityAttribute("4CB34728-4CB2-453D-AF9-08A680F2D77F"),
   SysAppWorkspaceSecurityAttribute("BatchGroup")]
   ```

2. Build the menu item and workspace. To test your changes, sign in to mobile app by using a user account that doesn’t have access to the menu item.

**Override the `workspaceHidden` method**

You can also specify whether the workspace is hidden or shown, based on parameters. By overriding the `workspaceHidden` method, you enable your code to control the visibility of the workspace, as shown in the following code example.

```csharp
/// <summary>
/// Used to control if a workspace is hidden or not
/// </summary>
/// <returns>Returns true if the workspace is hidden; otherwise false</returns>
public boolean workspaceHidden()
{
    boolean ret;
    ret = super();
    return ret;
}
```

**Add a menu item and override the `workspaceHidden` method**

You can use both the preceding methods in your app. The menu item provides a security check, and the `workspaceHidden` method contains additional logic that is related to the visibility of the workspace.
You can use workspace classes in several ways to provide localization support to workspaces.

**Use config objects to pass localized labels**

A config object can be added to the workspace metadata when it's requested by the mobile app. Later, the config object can be used to provide localization support.

For example, the following workspace requires localized labels for the `pageLink` control that you added via the business logic.

The business logic for the app contains a call to the `addLink` method, as shown in the following illustration. This `addLink` method adds a link to the `Rentals` page for the current customer. In this case, the label for the link is `Rentals`. However, because there isn't a localized label for the link, the link always appears as `Rentals`.

```javascript
```
To use a config object to provide localized labels, follow these steps.

1. Create a config class that contains the fields for the labels. One field, `rentalLinkLabel`, is added to the class that will contain the label for the `pageLink` control. The config class must be a data contract class.

```csharp
[DataContractAttribute("ReservationManagementLabels")]
class ReservationManagementWorkspaceLabels
{
    str rentalLinkLabel;
    [DataMemberAttribute("RentalLinkLabel")] public str rentalLinkLabel(str _rentalLinkLabel = rentalLinkLabel)
    {
        rentalLinkLabel = _rentalLinkLabel;
        return rentalLinkLabel;
    }
}
```

2. The config class is used by a workspace class for the workspace. The workspace class requires the `appId` value of the workspace. You can find the app ID in the App designer, as shown in the following illustration.

The following illustration shows what the workspace class looks like when the `appId` value is set on the attribute. The class also contains a method, `addConfig`, that sets a config object that contains the value for the label.
Use a workspace class to update the workspace title and description

A workspace class can be used to provide localized strings for the workspace title and description. If you don’t localize the title and description, the fields will be in the language that you implemented them in. In this example, we will localize a workspace where **MyWorkspace** is the title and **A sample workspace** is the description.
1. If you don't have a workspace class for your workspace, create a workspace class.

2. Override the `getWorkspaceMetadata` method to get the workspace metadata. You must have the workspace metadata to provide labels for the workspace title and description fields.

3. Use the `workspaceTitle` and `workspaceDescription` properties to set the workspace title and description from a label. In the following illustration, placeholders are assigned to the `workspaceTitle` and `workspaceDescription` properties.

```java
class WorkspaceInAOT extends SysAppWorkspace
{
    public SysAppWorkspaceMetadata getWorkspaceMetadata()
    {
        SysAppWorkspaceMetadata workspaceMetadata;
        workspaceMetadata = super();
        this.updateWorkspaceTitleAndDescription(workspaceMetadata);
        return workspaceMetadata;
    }

    private void updateWorkspaceTitleAndDescription(SysAppWorkspaceMetadata _workspaceMetadata)
    {
        _workspaceMetadata.workspaceTitle("@Client:Workspace");
        _workspaceMetadata.workspaceDescription("@ApplicationFoundation:FieldDescriptionsCaption");
    }
}
```

4. Build the workspace class.

5. Update the app list on the mobile client.

The following illustration shows the title and description on a phone that uses English and Danish.
Arbejdsområde
Beskrivelse af feltet
When you use the mobile app designer to select fields for actions, some properties can be inferred. These properties include the field length, the type, and whether the field is mandatory. The workspace classes can be used to update these properties. For example, you might want to specify that the Name field is mandatory when a customer record is created, as shown in the following images.
Follow these steps to make the Delivery terms field mandatory by using the workspace class.

1. Get the control name by using the app designer. In this example, the control name is DynamicDetail_DlvTerm.

2. Add the following code to set the Mandatory property for the control. This code uses the reflection-based setProperty method to set the Mandatory property.

   ```java
   public SysAppWorkspaceMetadata getWorkspaceMetadata()
   {
     SysAppWorkspaceMetadata appMetadata;
     appMetadata = super();
     var createCustAction = appMetadata.getAction("createCust");
     this.setCustAccountMandatory(createCustAction);
     return appMetadata;
   }
   private void setCustAccountMandatory(SysAppActionMetadata _createCustAction)
   {
     var custAccount = +createCustAction.getControl("DynamicDetail_DlvTerm");
     custAccount.setProperty("Mandatory", true);
   }
   ```

3. Build the solution, and then update the app metadata on the mobile app.

The Delivery terms field is now marked as Mandatory, as shown in the following illustration.
Name

Delivery terms

Mode of delivery
Class SysAppActionAttribute

SysAppActionAttribute used for decorating methods defining actions of workspace

### Methods

<table>
<thead>
<tr>
<th>METHOD NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>void</td>
<td>Creates a new instance of SysAppActionAttribute class</td>
</tr>
<tr>
<td>pageMethodName</td>
<td>str</td>
<td>Gets the get method name that forms the page under which this task resides</td>
</tr>
<tr>
<td>actionTitle</td>
<td>str</td>
<td>Gets the Action Title</td>
</tr>
<tr>
<td>actionDescription</td>
<td>str</td>
<td>Gets the Action Description</td>
</tr>
<tr>
<td>crudOperationType</td>
<td>SysAppCRUDOperation</td>
<td>Gets the Crud Operation Type like Create, Update, Delete</td>
</tr>
</tbody>
</table>

### Method new

Creates a new instance of SysAppActionAttribute class

```java
public void new ([str _actionTitle], [str _actionDescription], [SysAppCRUDOperation _crudOperationType], [str _pageMethodName])
```

#### Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_actionTitle</td>
<td>str</td>
<td>True</td>
<td>Action title</td>
</tr>
<tr>
<td>_actionDescription</td>
<td>str</td>
<td>True</td>
<td>Action description</td>
</tr>
<tr>
<td>_crudOperationType</td>
<td>SysAppCRUDOperation</td>
<td>True</td>
<td>CRUD operation like Create, Update, Delete</td>
</tr>
<tr>
<td>_pageMethodName</td>
<td>str</td>
<td>True</td>
<td>Name of the method constructing parent page</td>
</tr>
</tbody>
</table>

### Method pageMethodName

Gets the get method name that forms the page under which this task resides

```java
public str pageMethodName ()
```

### Return Value

...
The page method name that forms the page under which this task resides

**Method actionTitle**
Gets the Action Title

```java
public str actionTitle ()
```

**Return Value**
The action title

**Method actionDescription**
Gets the Action Description

```java
public str actionDescription ()
```

**Return Value**
The page description

**Method crudOperationType**
Gets the Crud Operation Type like Create, Update, Delete

```java
public SysAppCRUDOperation crudOperationType ()
```

**Return Value**
The Crud Operation Type like Create, Update, Delete

**Class SysAppActionMetadata**
This class can be used to access and update AX mobile workspace action metadata

**Methods**

<table>
<thead>
<tr>
<th>METHOD NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>void</td>
<td></td>
</tr>
<tr>
<td>getName</td>
<td>str</td>
<td>Returns the action name</td>
</tr>
<tr>
<td>actionTitle</td>
<td>str</td>
<td>Gets or sets the action title</td>
</tr>
<tr>
<td>actionDescription</td>
<td>str</td>
<td>Gets or sets the action description</td>
</tr>
<tr>
<td>actionHidden</td>
<td>boolean</td>
<td>Gets or sets whether action is hidden</td>
</tr>
<tr>
<td>actionOrder</td>
<td>int</td>
<td>Gets or sets the action order</td>
</tr>
<tr>
<td>getControl</td>
<td>SysAppControlMetadata</td>
<td>Returns the control on the current action having the provided control name</td>
</tr>
<tr>
<td>getControlEnumerator</td>
<td>MapEnumerator</td>
<td>Returns a map enumerator that can be used to enumerate action controls. Where Key is control name and value is of type SysAppControlMetadata</td>
</tr>
</tbody>
</table>
### Method `new`

```java
```

#### Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>

### Method `getActionName`

Returns the action name

```java
public str getActionName ()
```

#### Return Value

The action name

### Method `actionTitle`

Gets or sets the action title

```java
public str actionTitle ([str _actionTitle])
```

#### Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_actionTitle</td>
<td>str</td>
<td>True</td>
<td>The action title</td>
</tr>
</tbody>
</table>

#### Return Value

The action title

### Method `actionDescription`

Gets or sets the action description

```java
public str actionDescription ([str _actionDescription])
```

#### Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_actionDescription</td>
<td>str</td>
<td>True</td>
<td>The action description</td>
</tr>
</tbody>
</table>

#### Return Value

The action description

### Method `actionHidden`

Gets or sets whether action is hidden

```java
public boolean actionHidden ([boolean _actionHidden])
```
<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_actionHidden</td>
<td>boolean</td>
<td>True</td>
<td>Action hidden value</td>
</tr>
</tbody>
</table>

**Return Value**
True if the action is hidden; otherwise false

**Method actionOrder**
Gets or sets the action order

```java
public int actionOrder ([int _actionOrder])
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_actionOrder</td>
<td>int</td>
<td>True</td>
<td>The action order</td>
</tr>
</tbody>
</table>

**Return Value**
The action order

**Method getControl**
Returns the control on the current action having the provided control name

```java
public SysAppControlMetadata getControl (str _controlName)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_controlName</td>
<td>str</td>
<td>False</td>
<td>The control name that will be used to search for control</td>
</tr>
</tbody>
</table>

**Return Value**
An object of SysAppControlMetadata is returned if a control with the provided control name exist on the action; otherwise null

**Method getControlEnumerator**
Returns a map enumerator that can be used to enumerate action controls. Where Key is control name and value is of type SysAppControlMetadata

```java
public MapEnumerator getControlEnumerator ()
```

**Return Value**
A map enumerator

**Class SysAppAttributeHelper**
SysAppAttributeHelper class for fetching attributes from all the extended class

**Methods**
### Method getNameFromEntity

`getNameFromEntity` method retrieves a name from an entity.

- **Returns**: SysAttribute
- **Description**: gets attribute from class

#### Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Parameter Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_sysClass</td>
<td>SysDictClass</td>
<td>False</td>
<td>class for which attributes is required</td>
</tr>
<tr>
<td>_attributeType</td>
<td>SysAppAttributeType</td>
<td>False</td>
<td>Type of attribute like SysAppEntityAttribute</td>
</tr>
</tbody>
</table>

### Class SysAppCollectionAttribute

SysAppCollectionAttribute is used for decorating methods forming list control.

### Methods

- **Method new**
  - **Returns**: void
  - **Description**: Constructor

#### Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Parameter Type</th>
<th>Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_itemContractName</td>
<td>str</td>
<td>False</td>
<td>Data contract name of list item</td>
</tr>
<tr>
<td>_label</td>
<td>str</td>
<td>True</td>
<td>List control label</td>
</tr>
<tr>
<td>_relationshipName</td>
<td>str</td>
<td>True</td>
<td>Relationship name. By default the entity name of the list item is used as relationship name</td>
</tr>
</tbody>
</table>

### Class SysAppControlMetadata

Represents an X++ wrapper over the managed ControlMetadata object to facilitate, passing around the object as X++ object.

### Methods
<table>
<thead>
<tr>
<th>METHOD NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>void</td>
<td>Creates a new instance of the control metadata</td>
</tr>
<tr>
<td>getBaseLanguageId</td>
<td>str</td>
<td>Returns the base language ID for the app</td>
</tr>
<tr>
<td>getControlName</td>
<td>str</td>
<td>Returns the control name</td>
</tr>
<tr>
<td>controlLabel</td>
<td>str</td>
<td>Gets and sets the control label</td>
</tr>
<tr>
<td>controlHidden</td>
<td>boolean</td>
<td>Gets and sets whether the control is hidden</td>
</tr>
<tr>
<td>controlOrder</td>
<td>int</td>
<td>Gets or sets the control order</td>
</tr>
<tr>
<td>controlMandatory</td>
<td>boolean</td>
<td>Gets or sets the control mandatory</td>
</tr>
<tr>
<td>controlAllowNegative</td>
<td>boolean</td>
<td>Gets or sets the control allow negative</td>
</tr>
<tr>
<td>controlMaxLength</td>
<td>int</td>
<td>Gets or sets the control max length</td>
</tr>
<tr>
<td>getProperty</td>
<td>anytype</td>
<td>Gets the control property referenced by the key</td>
</tr>
<tr>
<td>setProperty</td>
<td>void</td>
<td>Sets the control property referenced by the key</td>
</tr>
</tbody>
</table>

**Method new**

Creates a new instance of the control metadata

```java
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_baseLanguageId</td>
<td>str</td>
<td>True</td>
<td>The base language</td>
</tr>
</tbody>
</table>

**Method getBaseLanguageId**

Returns the base language ID for the app

```java
public str getBaseLanguageId ()
```

**Return Value**

The base language ID

**Method getControlName**
Returns the control name

```
public str getControlName ()
```

**Return Value**
The control name

**Method controlLabel**
Gets and sets the control label

```
public str controlLabel ([str _controlLabel])
```

**Parameters**
<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_controlLabel</td>
<td>str</td>
<td>True</td>
<td>The control label</td>
</tr>
</tbody>
</table>

**Return Value**
The control label

**Method controlHidden**
Gets and sets whether the control is hidden

```
public boolean controlHidden ([boolean _controlHidden])
```

**Parameters**
<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_controlHidden</td>
<td>boolean</td>
<td>True</td>
<td>Control hidden value</td>
</tr>
</tbody>
</table>

**Return Value**
True if the control is hidden; otherwise false

**Method controlOrder**
Gets or sets the control order

```
public int controlOrder ([int _controlOrder])
```

**Parameters**
<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_controlOrder</td>
<td>int</td>
<td>True</td>
<td>The control order</td>
</tr>
</tbody>
</table>

**Return Value**
The control order

**Method controlMandatory**
Gets or sets the control mandatory

```
public boolean controlMandatory ([boolean _controlMandatory])
```
Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_controlMandatory</td>
<td>boolean</td>
<td>True</td>
<td>The control mandatory</td>
</tr>
</tbody>
</table>

Return Value

The control mandatory

Method controlAllowNegative

Gets or sets the control allow negative

```java
public boolean controlAllowNegative ([boolean _controlAllowNegative])
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_controlAllowNegative</td>
<td>boolean</td>
<td>True</td>
<td>The control allows negative</td>
</tr>
</tbody>
</table>

Return Value

The control allows negative

Method controlMaxLength

Gets or sets the control max length

```java
public int controlMaxLength ([int _controlMaxLength])
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_controlMaxLength</td>
<td>int</td>
<td>True</td>
<td>The control max length</td>
</tr>
</tbody>
</table>

Return Value

The control max length

Method getProperty

Gets the control property referenced by the key

```java
public anytype getProperty (str _key)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_key</td>
<td>str</td>
<td>False</td>
<td>The name of the control property</td>
</tr>
</tbody>
</table>

Return Value

The property value

Method setProperty

Sets the control property referenced by the key
```java
public void setProperty (str _key, anytype _value)
```

### Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_key</td>
<td>str</td>
<td>False</td>
<td>The name of the control property</td>
</tr>
<tr>
<td>_value</td>
<td>anytype</td>
<td>False</td>
<td>The value of the control property</td>
</tr>
</tbody>
</table>

## Class SysAppEntityAttribute

SysAppEntityAttribute used for decorating data contract entities

### Methods

<table>
<thead>
<tr>
<th>METHOD NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>void</td>
<td>Constructor</td>
</tr>
<tr>
<td>name</td>
<td>str</td>
<td>Gets the name of the entity</td>
</tr>
<tr>
<td>entityKey</td>
<td>str</td>
<td>Gets the entity key</td>
</tr>
</tbody>
</table>

### Method new

Constructor

```java
public void new (str _name, str _entityKey)
```

#### Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_name</td>
<td>str</td>
<td>False</td>
<td>Entity name</td>
</tr>
<tr>
<td>_entityKey</td>
<td>str</td>
<td>False</td>
<td>Name of the entity's key</td>
</tr>
</tbody>
</table>

### Method name

Gets the name of the entity

```java
public str name ()
```

#### Return Value

Name of the entity

### Method entityKey

Gets the entity key

```java
public str entityKey ()
```
Class SysAppEntityContext
SysAppEntityContext used for defining entity context

Methods

<table>
<thead>
<tr>
<th>METHOD NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>constructFromParams</td>
<td>SysAppEntityContext</td>
<td>Constructs SysAppEntityContext from entityName and entityId</td>
</tr>
<tr>
<td>constructFromBuffer</td>
<td>SysAppEntityContext</td>
<td>Constructs SysAppEntityContext from table buffer</td>
</tr>
<tr>
<td>entityName</td>
<td>str</td>
<td>Entity name on which filter applies</td>
</tr>
<tr>
<td>entityId</td>
<td>str</td>
<td>Field value on which filter applies</td>
</tr>
</tbody>
</table>

Method constructFromParams
Constructs SysAppEntityContext from entityName and entityId

```java
public SysAppEntityContext constructFromParams (str _entityName, str _entityId)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_entityName</td>
<td>str</td>
<td>False</td>
<td>Entity name</td>
</tr>
<tr>
<td>_entityId</td>
<td>str</td>
<td>False</td>
<td>Entity value</td>
</tr>
</tbody>
</table>

Return Value
Instance of SysAppEntityContext

Method constructFromBuffer
Constructs SysAppEntityContext from table buffer

```java
public SysAppEntityContext constructFromBuffer (Common _tableBuffer)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_tableBuffer</td>
<td>Common</td>
<td>False</td>
<td>table buffer forming the entity</td>
</tr>
</tbody>
</table>

Return Value
Instance of SysAppEntityContext

Method entityName
Entity name on which filter applies
public str entityName ([str _entityName])

Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_entityName</td>
<td>str</td>
<td>True</td>
<td>Entity name</td>
</tr>
</tbody>
</table>

Return Value

Entity name

Method entityId

Field value on which filter applies

public str entityId ([str _entityId])

Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_entityId</td>
<td>str</td>
<td>True</td>
<td>Entity value</td>
</tr>
</tbody>
</table>

Return Value

Entity value

Class SysAppFieldAttribute

SysAppFieldAttribute used for decorating methods forming bound fields

Methods

<table>
<thead>
<tr>
<th>METHOD NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>void</td>
<td>Creates a new instance of SysAppFieldAttribute class</td>
</tr>
</tbody>
</table>

Method new

Creates a new instance of SysAppFieldAttribute class

public void new (str _fieldName, str _label)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_fieldName</td>
<td>str</td>
<td>False</td>
<td>Entity field name with which control is bound</td>
</tr>
<tr>
<td>_label</td>
<td>str</td>
<td>False</td>
<td>Control label</td>
</tr>
</tbody>
</table>

Class SysAppFieldMultiSelectHelper

A helper class to provide helper methods for multi select scenarios used with D365 mobile app
Methods

<table>
<thead>
<tr>
<th>METHOD NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>void</td>
<td>Creates a new instance of SysAppFieldMultiSelectHelper class</td>
</tr>
<tr>
<td>getSelectedRecIds</td>
<td>container</td>
<td>Returns a container of recIds of the records that were selected</td>
</tr>
<tr>
<td>getSelectedValues</td>
<td>container</td>
<td>Returns a container of selected values</td>
</tr>
<tr>
<td>getSelectedRecords</td>
<td>Common</td>
<td>Returns a buffer that contains all the selected records. The buffer is marked as temp. Later a while-Select can be used to iterate through all the records</td>
</tr>
<tr>
<td>setControlValue</td>
<td>void</td>
<td>A setter to set the multiselect control value</td>
</tr>
</tbody>
</table>

**Method new**

Creates a new instance of SysAppFieldMultiSelectHelper class

```java
public void new (TableId _multiSelectTableId, FieldId _valueFieldId, FormStringControl _multiSelectControl)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_multiSelectTableId</td>
<td>TableId</td>
<td>False</td>
<td>The backing tableId for the field</td>
</tr>
<tr>
<td>_valueFieldId</td>
<td>FieldId</td>
<td>False</td>
<td>The fieldId for the field that will be shown in the multi select control</td>
</tr>
<tr>
<td>_multiSelectControl</td>
<td>FormStringControl</td>
<td>False</td>
<td>The string control that will be the multi select control</td>
</tr>
</tbody>
</table>

**Method getSelectedRecIds**

Returns a container of recIds of the records that were selected

```java
public container getSelectedRecIds ()
```

**Return Value**

A container of recOds for the records that were selected

**Method getSelectedValues**

Returns a container of selected values

```java
public container getSelectedValues ()
```

**Return Value**

A container of selected values
**Method getSelectedRecords**

Returns a buffer that contains all the selected records. The buffer is marked as temp. Later a while-Select can be used to iterate through all the records.

```java
public Common getSelectedRecords ()
```

**Return Value**

A buffer containing all the selected records.

**Method setControlValue**

A setter to set the multi select control value.

```java
public void setControlValue (str _value)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_value</td>
<td>str</td>
<td>False</td>
<td>A colon separated value that will be used by SysAppFieldMultiSelectHelper</td>
</tr>
</tbody>
</table>

**Class SysAppFilterContext**

SysAppFilterContext class that holds context values.

**Methods**

<table>
<thead>
<tr>
<th>METHOD NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>entityName</td>
<td>str</td>
<td>Entity name on which filter applies</td>
</tr>
<tr>
<td>filterFieldName</td>
<td>str</td>
<td>Field name on which filter applies</td>
</tr>
<tr>
<td>filterFieldValueList</td>
<td>List</td>
<td>Gets the list of filter field values based on which filter happens</td>
</tr>
<tr>
<td>operator</td>
<td>str</td>
<td>Operator based on which result will be fetched</td>
</tr>
<tr>
<td>addFilterFieldValue</td>
<td>void</td>
<td>Adds filter field value</td>
</tr>
</tbody>
</table>

**Method entityName**

Entity name on which filter applies

```java
public str entityName ([str _entityName])
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_entityName</td>
<td>str</td>
<td>True</td>
<td>Entity name on which filter applies</td>
</tr>
</tbody>
</table>

**Return Value**  
Entity name on which filter applies

**Method filterFieldName**  
Field name on which filter applies

```java
public str filterFieldName ([str _filterFieldName])
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_filterFieldName</td>
<td>str</td>
<td>True</td>
<td>Field name on which filter applies</td>
</tr>
</tbody>
</table>

**Return Value**  
Field name on which filter applies

**Method filterFieldValueList**  
Gets the list of filter field values based on which filter happens

```java
public List filterFieldValueList ()
```

**Return Value**  
List of filter field values based on which filter happens

**Method operator**  
Operator based on which result will be fetched

```java
public str operator ([str _operator])
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_operator</td>
<td>str</td>
<td>True</td>
<td>Operator based on which result will be fetched</td>
</tr>
</tbody>
</table>

**Return Value**  
Operator based on which result will be fetched

**Method addFilterFieldValue**  
Adds filter field value

```java
public void addFilterFieldValue ( _filterFieldValuelist)
```

**Parameters**
## Class SysAppLookUpAttribute

SysAppPageAttribute used for decorating pages that are also lookup pages

### Methods

<table>
<thead>
<tr>
<th>METHOD NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>void</td>
<td>Creates a new instance of SysAppLookUpAttribute class</td>
</tr>
<tr>
<td>displayFieldName</td>
<td>str</td>
<td>Gets the display field name of lookup control</td>
</tr>
<tr>
<td>valueFieldName</td>
<td>str</td>
<td>Gets the value field name of lookup control</td>
</tr>
</tbody>
</table>

### Method new

Creates a new instance of SysAppLookUpAttribute class

```java
public void new (str _displayFieldName, str _valueFieldName)
```

### Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_displayFieldName</td>
<td>str</td>
<td>False</td>
<td>Lookup display field. Name of any control of lookup page</td>
</tr>
<tr>
<td>_valueFieldName</td>
<td>str</td>
<td>False</td>
<td>Lookup value field. Name of control formed by root data contract constructing lookup page</td>
</tr>
</tbody>
</table>

### Method displayFieldName

Gets the display field name of lookup control

```java
public str displayFieldName ()
```

### Return Value

The display field name of lookup control

### Method valueFieldName

Gets the value field name of lookup control

```java
public str valueFieldName ()
```

### Return Value

The value field name of lookup control
The value field name of lookup control

Class SysAppLookupFieldAttribute
SysAppLookupFieldAttribute used for decorating look up fields of action

Methods

<table>
<thead>
<tr>
<th>METHOD_NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>void</td>
<td>Creates a new instance of SysAppLookupFieldAttribute class</td>
</tr>
<tr>
<td>entityName</td>
<td>str</td>
<td>Gets the name of the entity with which lookup page is related</td>
</tr>
</tbody>
</table>

Method new
Creates a new instance of SysAppLookupFieldAttribute class

```java
public void new (_name)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_name</td>
<td>False</td>
<td>Name of the entity with which lookup page is related</td>
<td></td>
</tr>
</tbody>
</table>

Method entityName
Gets the name of the entity with which lookup page is related

```java
public str entityName ()
```

Return Value
Name of the entity

Class SysAppPageAttribute
SysAppPageAttribute used for decorating methods defining page of workspace

Methods

<table>
<thead>
<tr>
<th>METHOD_NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>void</td>
<td>Creates a new instance of SysAppPageAttribute class</td>
</tr>
<tr>
<td>pageTitle</td>
<td>str</td>
<td>Gets the Page Title of the page</td>
</tr>
<tr>
<td>pageDescription</td>
<td>str</td>
<td>Gets the Page Description</td>
</tr>
</tbody>
</table>

Method new
Creates a new instance of SysAppPageAttribute class
public void new ([str _pageTitle], [str _pageDescription])

### Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_pageTitle</td>
<td>str</td>
<td>True</td>
<td>The page title</td>
</tr>
<tr>
<td>_pageDescription</td>
<td>str</td>
<td>True</td>
<td>The page description</td>
</tr>
</tbody>
</table>

### Method pageTitle

Gets the Page Title of the page

```java
public str pageTitle ()
```

**Return Value**

The page title

### Method pageDescription

Gets the Page Description

```java
public str pageDescription ()
```

**Return Value**

The page description

### Class SysAppPageMetadata

This class can be used to access and update AX mobile workspace page metadata

### Methods

<table>
<thead>
<tr>
<th>METHOD NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>void</td>
<td></td>
</tr>
<tr>
<td>getPageName</td>
<td>str</td>
<td></td>
</tr>
<tr>
<td>pageTitle</td>
<td>str</td>
<td>Gets and sets the page title</td>
</tr>
<tr>
<td>pageDescription</td>
<td>str</td>
<td>Gets or sets the page description</td>
</tr>
<tr>
<td>pageHidden</td>
<td>boolean</td>
<td>Gets and sets whether the page is hidden in the workspace</td>
</tr>
<tr>
<td>pageOrder</td>
<td>int</td>
<td>Gets or sets the page order</td>
</tr>
<tr>
<td>getControl</td>
<td>SysAppControlMetadata</td>
<td>Returns the control on the current page having the provided control name</td>
</tr>
</tbody>
</table>
### Method new

```java
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>

### Method getPageName

Returns the page name

```java
public str getPageName ()
```

**Return Value**

The page name

### Method pageTitle

Gets and sets the page title

```java
public str pageTitle ([str _pageTitle])
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_pageTitle</td>
<td>str</td>
<td>True</td>
<td>The page title</td>
</tr>
</tbody>
</table>

**Return Value**

The page title

### Method pageDescription

Gets or sets the page description

```java
public str pageDescription ([str _pageDescription])
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_pageDescription</td>
<td>str</td>
<td>True</td>
<td>The page description</td>
</tr>
</tbody>
</table>

**Return Value**

The page description
**Method pageHidden**

Gets and sets whether the page is hidden in the workspace

```java
public boolean pageHidden (boolean _pageHidden)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_pageHidden</td>
<td>boolean</td>
<td>True</td>
<td>page hidden value</td>
</tr>
</tbody>
</table>

**Return Value**

True if the current page is hidden in workspace; otherwise false

**Method pageOrder**

Gets or sets the page order

```java
public int pageOrder (int _pageOrder)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_pageOrder</td>
<td>int</td>
<td>True</td>
<td>The page order</td>
</tr>
</tbody>
</table>

**Return Value**

The page order

**Method getControl**

Returns the control on the current page having the provided control name

```java
public SysAppControlMetadata getControl (str _controlName)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_controlName</td>
<td>str</td>
<td>False</td>
<td>The control name that will be used to search for control</td>
</tr>
</tbody>
</table>

**Return Value**

An object of SysAppControlMetadata is returned if a control with the provided control name exist on the page; otherwise null

**Method getControlEnumerator**

Returns a map enumerator that can be used to enumerate page controls. Where Key is control name and value is of type SysAppControlMetadata

```java
public MapEnumerator getControlEnumerator ()
```

**Return Value**

A map enumerator
Class SysAppProjectionAttribute

SysAppProjectionAttribute used for decorating methods forming unbound fields

Methods

<table>
<thead>
<tr>
<th>METHOD NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>void</td>
<td>Creates a new instance of SysAppControlMetadataAttributes class</td>
</tr>
</tbody>
</table>

Method new

Creates a new instance of SysAppControlMetadataAttributes class

```java
public void new (str _label)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_label</td>
<td>str</td>
<td>False</td>
<td>Control label</td>
</tr>
</tbody>
</table>

Class SysAppRelationalAttribute

SysAppRelationalAttribute used for decorating reference controls

Methods

<table>
<thead>
<tr>
<th>METHOD NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>void</td>
<td>Constructor</td>
</tr>
</tbody>
</table>

Method new

Constructor

```java
public void new ([str _name])
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_name</td>
<td>str</td>
<td>True</td>
<td>Property name of the referenced entity</td>
</tr>
</tbody>
</table>

Class SysAppRequestParams

Request class for X++ methods generating details and action pages

Methods

<table>
<thead>
<tr>
<th>METHOD NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>entityContext</td>
<td>SysAppEntityContext</td>
<td>Entity context of the request</td>
</tr>
</tbody>
</table>
Method entityContext

Entity context of the request

```java
public SysAppEntityContext entityContext ([SysAppEntityContext _entityContext])
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_entityContext</td>
<td>SysAppEntityContext</td>
<td>True</td>
<td>Entity context of the request</td>
</tr>
</tbody>
</table>

**Return Value**

Entity context of the request

Method filterContext

List of SysAppFilterContext for filter contexts

```java
public List filterContext ([List _filterContext])
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_filterContext</td>
<td>List</td>
<td>True</td>
<td>List of SysAppFilterContext for filter contexts of page</td>
</tr>
</tbody>
</table>

**Return Value**

List of SysAppFilterContext for filter contexts of page

Class SysAppResponse

SysAppResponse class. This class holds the response object for generated pages and actions

**Methods**

<table>
<thead>
<tr>
<th>METHOD NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>void</td>
<td></td>
</tr>
<tr>
<td>jobId</td>
<td>str</td>
<td>Job ID of the request</td>
</tr>
<tr>
<td>failedInAppCall</td>
<td>boolean</td>
<td>Data of the page</td>
</tr>
<tr>
<td>commits</td>
<td>List</td>
<td>Commits after task is completed</td>
</tr>
<tr>
<td>METHOD NAME</td>
<td>RETURNS</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>messages</td>
<td>List</td>
<td>Job ID of the request</td>
</tr>
<tr>
<td>addMessage</td>
<td>void</td>
<td>Adds message</td>
</tr>
<tr>
<td>addCommit</td>
<td>void</td>
<td>Adds commits</td>
</tr>
</tbody>
</table>

**Method new**

```java
public void new ()
```

**Method jobId**

Job ID of the request

```java
public str jobId ()
```

**Return Value**

jobId of the request

**Method data**

Data of the page

```java
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>

**Return Value**

Data of the page

**Method failedInAppCall**

Data of the page

```java
public boolean failedInAppCall ([boolean _failedInAppCall])
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_failedInAppCall</td>
<td>boolean</td>
<td>True</td>
<td>Sets to true if it fails in calling application code</td>
</tr>
</tbody>
</table>

**Return Value**

True when fails in calling application code

**Method commits**
Commits after task is completed

```java
public List commits ()
```

**Return Value**
Commits after task is completed

**Method messages**
Job ID of the request

```java
public List messages ()
```

**Return Value**
Messages after task is completed

**Method addMessage**
Adds message

```java
public void addMessage (SysAppResponseMessage _message)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_message</td>
<td>SysAppResponseMessage</td>
<td>False</td>
<td>message as SysAppResponseMessage object</td>
</tr>
</tbody>
</table>

**Method addCommit**
Adds commits

```java
public void addCommit (SysAppEntityContext _entityContext)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_entityContext</td>
<td>SysAppEntityContext</td>
<td>False</td>
<td>Entity context containing entity name and entity ID of the entity that is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>committed</td>
</tr>
</tbody>
</table>

**Class SysAppResponseMessage**
SysAppResponseMessage class for response messages

**Methods**

<table>
<thead>
<tr>
<th>METHOD NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>void</td>
<td>Creates a new instance of SysAppResponseMessage class</td>
</tr>
</tbody>
</table>
Method new
Creates a new instance of SysAppResponseMessage class

```java
public void new (str _text, [SysAppMessage _type])
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_text</td>
<td>str</td>
<td>False</td>
<td>The message text</td>
</tr>
<tr>
<td>_type</td>
<td>SysAppMessageType</td>
<td>True</td>
<td>Message Type: info, error, warning</td>
</tr>
</tbody>
</table>

Method text
Gets the message text

```java
public str text ()
```

Return Value
The message text

Method type
Gets the message type: info, error, warning

```java
public SysAppMessageType type ()
```

Return Value
The message type: info, error, warning

Class SysAppSecurityAttribute

SysAppSecurityAttribute used for decorating methods forming pages and actions. specifies security attribute of page or action

Methods

<table>
<thead>
<tr>
<th>METHOD NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>void</td>
<td>Creates a new instance of SysAppSecurityAttribute class. This will help in checking if the user logged in has access to the specified menu item and menu item type</td>
</tr>
<tr>
<td>menuItemType</td>
<td>MenuItemType</td>
<td>Gets the Menu Item Type of the page</td>
</tr>
</tbody>
</table>
Method `new`  
Creates a new instance of `SysAppSecurityAttribute` class. This will help in checking if the user logged in has access to the specified menu item and menu item type

```java
public void new ([MenuItemName _menuItemName], [MenuItemType _menuItemType])
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_menuItemName</td>
<td>MenuItemName</td>
<td>True</td>
<td>Menu Item Name of the page</td>
</tr>
<tr>
<td>_menuItemType</td>
<td>MenuItemType</td>
<td>True</td>
<td>Menu Item Type of the page like action, display, or output</td>
</tr>
</tbody>
</table>

Method `menuItemType`  
Gets the Menu Item Type of the page

```java
public MenuItemType menuItemType ()
```

Return Value  
Menu Item Type of the page

Method `menuItemName`  
Gets the Menu Item Name of the page

```java
public MenuItemName menuItemName ()
```

Return Value  
Menu Item Name of the page

**Class SysAppWorkspace**

This is the base class of mobile workspace. Mobile workspace classes need to extend from this class

**Methods**

<table>
<thead>
<tr>
<th>METHOD NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>getEnumValues</td>
<td>List</td>
<td>Called during workspace initialization. Can be used to modify the enum values that are returned to AX mobile</td>
</tr>
<tr>
<td>getWorkspaceMetadata</td>
<td>SysAppWorkspaceMetadata</td>
<td>Called during workspace initialization. Can be used to modify the workspace metadata</td>
</tr>
<tr>
<td>METHOD NAME</td>
<td>RETURNS</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>onBeginAppJob</td>
<td>void</td>
<td>Called before the start of execution of AX mobile job</td>
</tr>
<tr>
<td>onEndAppJob</td>
<td>void</td>
<td>Called after the end of execution of AX mobile job</td>
</tr>
<tr>
<td>workspaceHidden</td>
<td>boolean</td>
<td>Can be used to control whether the workspace is hidden or not. Checks that the current user has access menu item specified by SysAppWorkspaceSecurityAttribute on the workspace class. If the attribute is not specified on the class then it always returns false</td>
</tr>
</tbody>
</table>

**Method getEnumValues**

Called during workspace initialization. Can be used to modify the enum values that are returned to AX mobile

```java
public List getEnumValues (EnumName _enumName)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_enumName</td>
<td>EnumName</td>
<td>False</td>
<td>The enum name</td>
</tr>
</tbody>
</table>

**Return Value**

A list of enum value

**Method getWorkspaceMetadata**

Called during workspace initialization. Can be used to modify the workspace metadata

```java
public SysAppWorkspaceMetadata getWorkspaceMetadata ()
```

**Return Value**

An object representing the workspace metadata

**Method onBeginAppJob**

Called before the start of execution of AX mobile job

```java
public void onBeginAppJob (SysAppJobRequest _sysAppJobRequest)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_sysAppJobRequest</td>
<td>SysAppJobRequest</td>
<td>False</td>
<td>A class containing job request parameters</td>
</tr>
</tbody>
</table>

**Method onEndAppJob**

Called after the end of execution of AX mobile job
public void onEndAppJob (SysAppJobResponse _sysAppJobResponse)

Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_sysAppJobResponse</td>
<td>SysAppJobResponse</td>
<td>False</td>
<td>A class containing job response parameters</td>
</tr>
</tbody>
</table>

Method `workspaceHidden`

Can be used to control whether the workspace is hidden or not. Checks that the current user has access menu item specified by `SysAppWorkspaceSecurityAttribute` on the workspace class. If the attribute is not specified on the class then it always returns false.

```java
public boolean workspaceHidden ()
```

Return Value

Returns true if the workspace is hidden otherwise false.

Class `SysAppWorkspaceAttribute`

Applied on classes that are extended from `SysAppWorkspace`

Methods

<table>
<thead>
<tr>
<th>METHOD NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>new</code></td>
<td>void</td>
<td>Creates a new instance of <code>SysAppWorkspaceAttribute</code> class</td>
</tr>
<tr>
<td>AppId</td>
<td>str</td>
<td>Gets or sets the AppId of the workspace</td>
</tr>
<tr>
<td>AppResourceName</td>
<td>str</td>
<td>Gets or sets the AOT Resource name that contains the workspace</td>
</tr>
<tr>
<td>WorkspaceHidden</td>
<td>boolean</td>
<td>Gets or sets if the workspace is hidden from designer</td>
</tr>
</tbody>
</table>

Method `new`

Creates a new instance of `SysAppWorkspaceAttribute` class

```java
public void new (str _appId, [str _appResourceName], [boolean _workspaceHidden])
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_appId</td>
<td>str</td>
<td>False</td>
<td>The appId of the workspace</td>
</tr>
<tr>
<td>_appResourceName</td>
<td>str</td>
<td>True</td>
<td>The AOT resource name that contains the workspace</td>
</tr>
</tbody>
</table>
### Method AppId
Gets or sets the AppId of the workspace

```java
public str AppId ([str _appId])
```

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_appId</td>
<td>str</td>
<td>True</td>
<td>The AppId of the workspace</td>
</tr>
</tbody>
</table>

**Return Value**
The AppId of the workspace

### Method AppResourceName
Gets or sets the AOT Resource name that contains the workspace

```java
public str AppResourceName ([str _appResourceName])
```

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_appResourceName</td>
<td>str</td>
<td>True</td>
<td>The AOT Resource name that contains the workspace</td>
</tr>
</tbody>
</table>

**Return Value**
The AOT Resource name that contains the workspace

### Method WorkspaceHidden
Gets or sets if the workspace is hidden from designer

```java
public boolean WorkspaceHidden ([boolean _workspaceHidden])
```

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_workspaceHidden</td>
<td>boolean</td>
<td>True</td>
<td>The workspace hidden</td>
</tr>
</tbody>
</table>

**Return Value**
Whether the workspace is hidden from designer or not

### Class SysAppWorkspaceMetadata
This class can be used to access and update metadata of an AX mobile workspace

### Methods
<table>
<thead>
<tr>
<th>METHOD NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>void</td>
<td></td>
</tr>
<tr>
<td>addConfig</td>
<td>void</td>
<td>Adds a custom config to the mobile workspace metadata</td>
</tr>
<tr>
<td>getPage</td>
<td>SysAppPageMetadata</td>
<td>Returns the page with the pageName provided</td>
</tr>
<tr>
<td>getPageEnumerator</td>
<td>MapEnumerator</td>
<td>Returns a map enumerator that can be used to enumerate workspace pages. Where key is page name and value is of type SysAppPageMetadata</td>
</tr>
<tr>
<td>getPageNameForRecordingId</td>
<td>str</td>
<td>Returns a pageName if the provided recordingId is used by a workspace page</td>
</tr>
<tr>
<td>getPageNameForRecordingId</td>
<td>str</td>
<td>Returns an actionName if the provided recordingId is used by a workspace action</td>
</tr>
<tr>
<td>workspaceTitle</td>
<td>str</td>
<td>Gets and sets the workspace title</td>
</tr>
<tr>
<td>workspaceDescription</td>
<td>str</td>
<td>Gets and sets the workspace description</td>
</tr>
</tbody>
</table>

**Method new**

```
public void new (str _appId, [SysAppWorkspaceAttribute _attribute])
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_appId</td>
<td>str</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>_attribute</td>
<td>SysAppWorkspaceAttribute</td>
<td>True</td>
<td></td>
</tr>
</tbody>
</table>

**Method addConfig**

Adds a custom config to the mobile workspace metadata

```
public void addConfig (str _configName, object _configValue)
```

**Parameters**
Method getPage

Returns the page with the pageName provided

```java
public SysAppPageMetadata getPage (str _pageName)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_pageName</td>
<td>str</td>
<td>False</td>
<td>A page name</td>
</tr>
</tbody>
</table>

Return Value

Returns the pageMetadata if a page with the provided name exists; otherwise null

Method getPageEnumerator

Returns a map enumerator that can be used to enumerate workspace pages. Where key is page name and value is of type SysAppPageMetadata

```java
public MapEnumerator getPageEnumerator ()
```

Return Value

A map enumerator

Method getAction

Returns the action with the actionName provided

```java
public SysAppActionMetadata getAction (str _actionName)
```

Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_actionName</td>
<td>str</td>
<td>False</td>
<td>An action name</td>
</tr>
</tbody>
</table>

Return Value

Returns the ActionMetadata if an action with the provided name exists; otherwise null

Method getActionEnumerator

Returns a map enumerator that can be used to enumerate workspace actions. Where key is action name and value is of type SysAppActionMetadata

```java
public MapEnumerator getActionEnumerator ()
```

Return Value

A map enumerator
**Method getPageNameForRecordingId**

Returns a pageName if the provided recordingId is used by a workspace page

```java
public str getPageNameForRecordingId (str _recordingId)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_recordingId</td>
<td>str</td>
<td>False</td>
<td>A recordingId</td>
</tr>
</tbody>
</table>

**Return Value**

A page name if the supplied recordingId is used by a workspace page; otherwise empty string

**Method getActionNameForRecordingId**

Returns a actionName if the provided recordingId is used by a workspace action

```java
public str getActionNameForRecordingId (str _recordingId)
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_recordingId</td>
<td>str</td>
<td>False</td>
<td>A recordingId</td>
</tr>
</tbody>
</table>

**Return Value**

An action name if the supplied recordingId is used by a workspace action; otherwise empty string

**Method workspaceTitle**

Gets and sets the workspace title

```java
public str workspaceTitle ([str _workspaceTitle])
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_workspaceTitle</td>
<td>str</td>
<td>True</td>
<td>The workspace title</td>
</tr>
</tbody>
</table>

**Return Value**

The workspace title

**Method workspaceDescription**

Gets and sets the workspace description

```java
public str workspaceDescription ([str _workspaceDescription])
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_workspaceDescription</td>
<td>str</td>
<td>True</td>
<td>The workspace description</td>
</tr>
</tbody>
</table>

**Return Value**
The workspace description

**Class SysAppWorkspaceSecurityAttribute**

Controls the visibility based of workspace based on the menu item tied to this attribute

**Methods**

<table>
<thead>
<tr>
<th>METHOD NAME</th>
<th>RETURNS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>void</td>
<td>Creates a new instance of attribute</td>
</tr>
<tr>
<td>WorkspaceMenuItemName</td>
<td>MenuItemName</td>
<td>Gets or sets the workspace menuitem for the workspace security attribute</td>
</tr>
<tr>
<td>WorkspaceMenuItemType</td>
<td>MenuItemType</td>
<td>Gets or sets the workspace menu item type for the workspace security attribute</td>
</tr>
</tbody>
</table>

**Method new**

Creates a new instance of attribute

```java
public void new (MenuItemName _workspaceMenuItemName, [MenuItemType _workspaceMenuItemType])
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_workspaceMenuItemName</td>
<td>MenuItemName</td>
<td>False</td>
<td>The menu item name to which the workspace needs to be tied</td>
</tr>
<tr>
<td>_workspaceMenuItemType</td>
<td>MenuItemType</td>
<td>True</td>
<td>The menu item type</td>
</tr>
</tbody>
</table>

**Method WorkspaceMenuItemName**

Gets or sets the workspace menuitem for the workspace security attribute

```java
public MenuItemName WorkspaceMenuItemName ([MenuItemName _workspaceMenuItemName])
```

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_workspaceMenuItemName</td>
<td>MenuItemName</td>
<td>True</td>
<td>The workspace menu item for the workspace security attribute</td>
</tr>
</tbody>
</table>

**Return Value**

The workspace menu item for the workspace security attribute

**Method WorkspaceMenuItemType**

Gets or sets the workspace menu item type for the workspace security attribute

```java
public MenuItemType WorkspaceMenuItemType ([MenuItemType _workspaceMenuItemType])
```
## Parameters

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>PARAMETER TYPE</th>
<th>OPTIONAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>_workspaceMenuItemType</td>
<td>MenuItemType</td>
<td>True</td>
<td>The workspace menu item type for the workspacesecurity attribute</td>
</tr>
</tbody>
</table>

## Return Value

The workspace menu item type for the workspace security attribute.
This topic provides an overview of the application programming interfaces (APIs) for client-side design and includes recommendations for using them.

**Terminology**

The following list includes some frequently used terms that apply to the client-side design APIs.

- **Design** – A property that can optionally be specified on a Page, Action, or other component object to override its default design.

- **Component** – A component can be one of four types:
  - **Block container** (default) – A container that has CSS block behaviors. (In other words, the container is equivalent to an element that has a CSS `display: block` style declaration.)
  - **Flex container** – A container that has CSS flex behaviors. (In other words, the container is equivalent to an element that has a CSS `display: flex` style declaration.)
  - **Control reference** – A component that refers to a control that exists in the static metadata (XML) of the Page or Action.
  - **New control** – A component that instantiates a new control. (In other words, the control doesn’t already exist in the static metadata [XML] of the Page or Action.)

  A component is represented in the design by a JavaScript Object Notation (JSON) object, and the properties of this JSON object represent the properties of the component. Almost every JSON object in the design property hierarchy is a component.

- **Item** – A component that is nested in a container.

- **Property** – Several types of properties can be set on a component:
  - Container-specific
  - Item-specific
  - Control-specific
  - List-specific
  - Generic (non-specific)

  Properties are specified as key-value pairs on the component’s JSON object. The properties that are applicable depend on the type of component that the property is applied to.

  For properties that have a predefined list of possible values, the first value that is shown in the documentation is the property’s default value. In most cases, if you don’t specify a property at all (that is, if you omit the property from the JSON object), the property behaves as if you had set its default value.

  Generic properties can be applied to all component types.

  When you specify properties, follow these guidelines:
You should not enclose property names in quotation marks.

You must enclose all property values in double quotation marks, unless the documentation specifies otherwise.

- **Inheritance** – If a color, font size, or font weight is applied to a control, all descendant controls inherit the same property, unless they are reassigned. If padding is applied to a control, it’s inherited by the item (non-container) descendants of the control. No other properties are inherited.

### Using design APIs

The following code is a modified segment of business logic code from a Reservation Management example. Specifically, this code is from a variable that specifies the design for a reservation details page. Comments are included in the code to highlight a few possibilities.

```plaintext
NOTE

Any color, font size, or font weight that is applied to a control is also applied to all children of that control. Padding is inherited by non-container children. No other properties are inherited. Containers include lists, pages, groups, and parts.

After a control is created that doesn't have any children or items, the control name just has to be written in quotation marks (see `FMCustomer_FullName` in the following code). However, if any customization will be applied to that control, the code must be blocked, and the **name** label must be used (see `FMCustomer_Image` in the following code).
```
The following illustration shows the customer image, customer name, font, background color, and so on, that preceding code produces.
Status: Ready for pickup
Id: 000033

Pickup: 2/8/2017, 7:17:39 PM
Return: 4/8/2017, 12:00:00 AM

Complete
Delete
Edit

Search by Description

Child Seat 10.00
Child Seat 10.00

Add charge

Vehicle: 2030.00
Sub-total: 2430.00
• Application
  • Application
  • ApplicationMetadata
  • main
• Control
  • Container
    • ContainerControl
    • ContainerControlDesign
    • ContainerControlMetadata
  • Field
    • Field
    • FieldDesign
    • FieldMetadata
  • File Uploader
    • FileUploader
    • FileUploaderDesign
    • FileUploaderMetadata
• Group
  • Group
    • GroupDesign
    • GroupMetadata
• Hyperlink
  • HyperLink
    • HyperLinkDesign
    • HyperLinkMetadata
• Image
  • Image
    • ImageDesign
    • ImageMetadata
- ImageStyleType
- Input
  - InputControl
  - InputControlDesign
  - InputControlMetadata
  - NumberSequenceConfig
- List
  - List
  - ListDesign
  - ListMetadata
  - Row
- Lookup
  - Lookup
  - LookupDesign
  - LookupMetadata
- Multi-Lookup
  - MultiLookup
  - MultiLookupDesign
  - MultiLookupMetadata
- Pagelink
  - PageLink
  - PageLinkDesign
  - PageLinkMetadata
- Part
  - Part
  - PartDesign
  - PartMetadata
- Value
  - GenericValue
  - Value
  - ValueDesign
  - ValueMetadata
- Control
ControlMetadata
ControlType

- Defers
  - all
  - defer
  - Deferred
  - reject
  - resolve

- Event
  - EventHook
  - IEventListener

- Page
  - CompleteEventArgs
  - Design
  - NavigationArgs
  - Page
  - PageMetadata
  - PageOptions
  - PageState
  - PageSubmitArgs
  - PageTarget

- Services
  - AsyncService
  - CacheService
  - DataService
  - ExpressionOperator
  - MetadataService
  - PageData
Represents the various high-level states the page can be in.

Index

Enumeration members
- error
- loaded
- loading
- offline
- refreshing

Enumeration members

error
error: 10 The page is currently in the error state, but can be refreshed in an attempt to get out of this state.

loaded
loaded: 3 The page is fully loaded and can be refreshed and, if possible, submitted.

loading
loading: 2 The page is currently being loaded.

offline
offline: 1 The page was loaded in the offline mode, thus not refreshable.

refreshing
refreshing: 4 The page is currently refreshing its data.
An application is a unit of runtime execution with sandboxing around concepts and data used inside of it. Each application consists of pages, actions, data queries, and logic that glue them together. An application is primarily described with a declarative metadata system, and can have an accompanying imperative extension model.

The imperative extension of the application is typically defined in a script module with a designated entry point, the `main function`, which allows the imperative logic to integrate with the application life cycle.

### Index

#### Types
- Application
- ApplicationMetadata

#### Functions
- main

### Types

**Application**

**Hierarchy**

Application

#### Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>minVersion</td>
<td>minVersion: string (optional)</td>
<td>An optional marker to indicate the minimum platform version required by this component. When this value is specified and the component tries to load in an older version of the platform, the corresponding workspace is not loaded and user is directed to install a newer version of the platform.</td>
</tr>
</tbody>
</table>

#### Methods

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>appInit</td>
<td>appInit(metadata: ApplicationMetadata): any</td>
<td>This method is invoked at the point the application is about to run, with the instance of the application metadata loaded. The metadata passed in can be still modified to change behaviors before this method returns.</td>
</tr>
</tbody>
</table>

**ApplicationMetadata**

**Hierarchy**

ApplicationMetadata
Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColorName</td>
<td>ColorName: string (optional)</td>
<td>The theme color of the application</td>
</tr>
<tr>
<td>Configs</td>
<td>Configs: [name: string]: any (optional)</td>
<td>An application can have a set of named config supplied by the author or the resource provider</td>
</tr>
<tr>
<td>Description</td>
<td>Description: string (optional)</td>
<td>The description of the application</td>
</tr>
<tr>
<td>IconName</td>
<td>IconName: string (optional)</td>
<td>The representative icon of the application</td>
</tr>
<tr>
<td>ID</td>
<td>ID: string</td>
<td>The unique identifier of the application</td>
</tr>
<tr>
<td>Title</td>
<td>Title: string</td>
<td>The title of the application</td>
</tr>
</tbody>
</table>

Functions

**main**

```plaintext
```

The main method of a business logic module. Each business logic module (as JavaScript file) must contain one main method. The method is invoked when the module is loaded and is being initialized. The method must return the component to run from this module.

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>metadataService</td>
<td>MetadataService</td>
<td></td>
</tr>
<tr>
<td>dataService</td>
<td>DataService</td>
<td></td>
</tr>
<tr>
<td>cacheService</td>
<td>CacheService</td>
<td></td>
</tr>
<tr>
<td>asyncService</td>
<td>AsyncService</td>
<td></td>
</tr>
</tbody>
</table>

Returns **Application**
Index

Types
- Deferred

Functions
- all
- defer
- reject
- resolve

Types

Deferred

Hierarchy
Deferred

Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>promise</td>
<td>promise: Promise &lt;T&gt;</td>
<td></td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>reject</td>
<td>reject(error?: any): void</td>
<td></td>
</tr>
<tr>
<td>resolve</td>
<td>resolve(value?: T</td>
<td>PromiseLike &lt;T&gt;): void</td>
</tr>
</tbody>
</table>

Functions

all

all(...args: any []): Promise <any []>

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>...args</td>
<td>any []</td>
<td></td>
</tr>
</tbody>
</table>

Returns Promise <any []>

defer
defer <T>(): Deferred <T>

Returns Deferred <T>
**reject**

reject(error?: any): Promise <any>

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>error?</td>
<td>any</td>
<td></td>
</tr>
</tbody>
</table>

**Returns Promise <any>**

**resolve**

resolve <T>(value?: T | PromiseLike <T>): Promise <T>

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>value?</td>
<td>T</td>
<td>PromiseLike &lt;T&gt;</td>
</tr>
</tbody>
</table>

**Returns Promise <T>**
Event module

Index

Types
- EventHook

Type aliases
- IEventListener

Types

EventHook

Hierarchy
EventHook

Methods

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>subscribe</td>
<td>subscribe(listener: IEventListener &lt;T&gt;): void</td>
<td>Subscribe a listener to this event.</td>
</tr>
<tr>
<td>unsubscribe</td>
<td>unsubscribe(listener: IEventListener &lt;T&gt;): void</td>
<td>Unsubscribe a listener from this event.</td>
</tr>
<tr>
<td>unsubscribeAll</td>
<td>unsubscribeAll(): void</td>
<td>Remove all listeners from this event.</td>
</tr>
</tbody>
</table>

Type aliases

IEventListener

IEventListener: function
Controls are what make up the content of a page.

Index

Types
- Control
- ControlMetadata

Type aliases
- ControlType

Types

Control

Hierarchy
Control
  - PageLink
  - ContainerControl
  - InputControl
  - Image

Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>container: boolean (optional)</td>
<td>True if the control is a container.</td>
</tr>
<tr>
<td>generic</td>
<td>generic: boolean (optional)</td>
<td></td>
</tr>
<tr>
<td>getDataSource</td>
<td>getDataSource: function(): any</td>
<td></td>
</tr>
<tr>
<td>hidden</td>
<td>hidden: boolean</td>
<td>True if the control is hidden.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>applyDesign</td>
<td>applyDesign(design: Design): void</td>
<td>Applies given design to the design on the control.</td>
</tr>
<tr>
<td>dataContext</td>
<td>dataContext(): any</td>
<td></td>
</tr>
<tr>
<td>getDesign</td>
<td>getDesign(): Design</td>
<td>Returns the design object of this control.</td>
</tr>
<tr>
<td>isEditable</td>
<td>isEditable(): boolean</td>
<td>Boolean indicating if the control is editable.</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>metadata</td>
<td>metadata(): ControlMetadata</td>
<td>Returns the metadata object of this control.</td>
</tr>
<tr>
<td>parent</td>
<td>parent(): Control</td>
<td>Page</td>
</tr>
<tr>
<td>root</td>
<td>root(): Page</td>
<td>Returns the root form instance (page) of this control.</td>
</tr>
</tbody>
</table>

**ControlMetadata**

**Hierarchy**

ControlMetadata → PageLinkMetadata → ContainerControlMetadata → InputControlMetadata → ImageMetadata

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoundEntity</td>
<td>BoundEntity: string (optional)</td>
<td>The entity to which the control is bound.</td>
</tr>
<tr>
<td>BoundField</td>
<td>BoundField: string (optional)</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Description: string (optional)</td>
<td>Description of the control.</td>
</tr>
<tr>
<td>Editable</td>
<td>Editable: boolean (optional)</td>
<td>Boolean indicating if the control is editable.</td>
</tr>
<tr>
<td>ExtType</td>
<td>ExtType: ControlType (optional)</td>
<td>The extended control type. For example, a control of type Input might have an extended type of Barcode.</td>
</tr>
<tr>
<td>HelpText</td>
<td>HelpText: string (optional)</td>
<td>The keyboard shortcut for a command. For example, &quot;(Shift+F5)&quot;</td>
</tr>
<tr>
<td>Hidden</td>
<td>Hidden: boolean (optional)</td>
<td>Boolean indicating if the control is hidden or not.</td>
</tr>
<tr>
<td>Id</td>
<td>Id: string (optional)</td>
<td>Identification string for a control.</td>
</tr>
<tr>
<td>Label</td>
<td>Label: string (optional)</td>
<td>Label for a control. For example, a control representing a person's first name might have a label &quot;First Name&quot;.</td>
</tr>
<tr>
<td>Name</td>
<td>Name: string (optional)</td>
<td>Name of a control.</td>
</tr>
<tr>
<td>Order</td>
<td>Order: number (optional)</td>
<td>Number indicating the order in which a control will appear on a page.</td>
</tr>
<tr>
<td>Type</td>
<td>Type: ControlType (optional)</td>
<td>String indicating the control type.</td>
</tr>
</tbody>
</table>
Type aliases

ControlType

ControlType: "FileUpload" | "Barcode" | "Input" | "MultilineInput" | "Navigation" | "Integer" | "Int64" | "Date" | "DateTime" | "ComboBox" | "Real" | "List" | "Lookup" | "MultiLookup" | "Navigation" | "Image" | "Group" | "Part" | "Calendar" | "HyperLink" | "Timer"

Controls must be assigned any of the types listed in ControlType.
A container control can contain any number of controls.

Index

Types
- ContainerControl
- ContainerControlDesign
- ContainerControlMetadata

Types

ContainerControl

Hierarchy
Control
  └ ContainerControl
    └ Group
    └ List
    └ Part

Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>container: boolean</td>
<td>True if the control is a container. Overrides Control.container</td>
</tr>
<tr>
<td>generic</td>
<td>generic: boolean (optional)</td>
<td>Inherited from Control.generic</td>
</tr>
<tr>
<td>getDataSource</td>
<td>getDataSource: function(): any</td>
<td>Inherited from Control.getDataSource</td>
</tr>
<tr>
<td>hidden</td>
<td>hidden: boolean</td>
<td>True if the control is hidden. Inherited from Control.hidden</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>applyDesign</td>
<td>applyDesign(design: Design): void</td>
<td>Applies given design to the design on the control. Inherited from Control.applyDesign</td>
</tr>
<tr>
<td>dataContext</td>
<td>dataContext(): any</td>
<td>Inherited from Control.dataContext</td>
</tr>
<tr>
<td>getControl</td>
<td>getControl(controlName: string): Control</td>
<td>Given the name of a control, returns the control instance.</td>
</tr>
<tr>
<td>getControlById</td>
<td>getControlById(id: string): Control</td>
<td>Given the ID of a control, returns the control instance.</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>getDesign</td>
<td>getDesign(): Design</td>
<td>Returns the design object of this control. Inherited from Control.getDesign</td>
</tr>
<tr>
<td>isEditable</td>
<td>isEditable(): boolean</td>
<td>Boolean indicating if the control is editable. Inherited from Control.isEditable</td>
</tr>
<tr>
<td>metadata</td>
<td>metadata(): ContainerControlMetadata</td>
<td>Returns the metadata object of this control. Overrides Control.metadata</td>
</tr>
<tr>
<td>parent</td>
<td>parent(): Control</td>
<td>Page</td>
</tr>
<tr>
<td>root</td>
<td>root(): Page</td>
<td>Returns the root form instance (page) of this control. Inherited from Control.root</td>
</tr>
</tbody>
</table>

**ContainerControlDesign**

Hierarchy

Design
- ContainerControlDesign
  - GroupDesign
  - ListDesign
  - PartDesign

Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>alignItems</td>
<td>alignItems: string (optional)</td>
<td>This property is an alias for the CSS property &quot;align-items&quot;. Inherited from Design.alignItems</td>
</tr>
<tr>
<td>alignSelf</td>
<td>alignSelf: string (optional)</td>
<td>Inherited from Design.alignSelf</td>
</tr>
<tr>
<td>allowScroll</td>
<td>allowScroll: string (optional)</td>
<td>True if the container will allow scrolling when its items do not fit into the container's available space. If a container has an item which may scroll, then set this property to false to prevent nested scrolling areas.</td>
</tr>
<tr>
<td>background</td>
<td>background: string (optional)</td>
<td>The background color of the container.</td>
</tr>
<tr>
<td>bindings</td>
<td>bindings: any (optional)</td>
<td>Inherited from Design.bindings</td>
</tr>
<tr>
<td>border</td>
<td>border: &quot;none&quot;</td>
<td>&quot;solid&quot;</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>color</td>
<td>color: string (optional)</td>
<td>The foreground color of the container. Inherited from Design.color</td>
</tr>
<tr>
<td>flexFlow</td>
<td>flexFlow: string (optional)</td>
<td>Specifying this property makes the component a flex container component. Inherited from Design.flexFlow</td>
</tr>
<tr>
<td>flexSize</td>
<td>flexSize: string (optional)</td>
<td>One number or two numbers written as a string. For example, &quot;(size to grow) ([(size-to-shrink)])&quot; to accommodate available space in the immediate flex container. Inherited from Design.flexSize</td>
</tr>
<tr>
<td>fontSize</td>
<td>fontSize: &quot;medium&quot;</td>
<td>&quot;xx-small&quot;</td>
</tr>
<tr>
<td>fontWeight</td>
<td>fontWeight: &quot;normal&quot;</td>
<td>&quot;bold&quot; (optional)</td>
</tr>
<tr>
<td>itemBorder</td>
<td>itemBorder: &quot;solid&quot;</td>
<td>&quot;none&quot; (optional)</td>
</tr>
<tr>
<td>items</td>
<td>items: string</td>
<td>Design [ ] (optional)</td>
</tr>
<tr>
<td>justifyItems</td>
<td>justifyItems: &quot;flex-start&quot;</td>
<td>&quot;flex-end&quot;</td>
</tr>
<tr>
<td>label</td>
<td>label: string (optional)</td>
<td>Inherited from Design.label</td>
</tr>
<tr>
<td>labelPosition</td>
<td>labelPosition: &quot;stacked&quot;</td>
<td>&quot;hidden&quot;</td>
</tr>
<tr>
<td>name</td>
<td>name: string (optional)</td>
<td>Inherited from Design.name</td>
</tr>
<tr>
<td>padding</td>
<td>padding: &quot;none&quot;</td>
<td>&quot;small&quot;</td>
</tr>
<tr>
<td>type</td>
<td>type: ControlType (optional)</td>
<td>The type of the control as a string. Inherited from Design.type</td>
</tr>
</tbody>
</table>

**ContainerControlMetadata**

**Hierarchy**

**ControlMetadata**

- ContainerControlMetadata
- GroupMetadata
- ListMetadata
<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoundEntity</td>
<td>BoundEntity: string (optional)</td>
<td>The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity</td>
</tr>
<tr>
<td>BoundField</td>
<td>BoundField: string (optional)</td>
<td>Inherited from ControlMetadata.BoundField</td>
</tr>
<tr>
<td>Description</td>
<td>Description: string (optional)</td>
<td>Description of the control. Inherited from ControlMetadata.Description</td>
</tr>
<tr>
<td>Editable</td>
<td>Editable: boolean (optional)</td>
<td>Boolean indicating if the control is editable. Inherited from ControlMetadataEditable</td>
</tr>
<tr>
<td>ExtType</td>
<td>ExtType: ControlType (optional)</td>
<td>The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType</td>
</tr>
<tr>
<td>HelpText</td>
<td>HelpText: string (optional)</td>
<td>The keyboard shortcut for a command. For example, “(Shift+F5)” Inherited from ControlMetadata.HelpText</td>
</tr>
<tr>
<td>Hidden</td>
<td>Hidden: boolean (optional)</td>
<td>Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden</td>
</tr>
<tr>
<td>Id</td>
<td>Id: string (optional)</td>
<td>Identification string for a control. Inherited from ControlMetadata.Id</td>
</tr>
<tr>
<td>Label</td>
<td>Label: string (optional)</td>
<td>Label for a control. For example, a control representing a person’s first name might have a label “First Name”. Inherited from ControlMetadata.Label</td>
</tr>
<tr>
<td>Name</td>
<td>Name: string (optional)</td>
<td>Name of a control. Inherited from ControlMetadata.Name</td>
</tr>
<tr>
<td>Order</td>
<td>Order: number (optional)</td>
<td>Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order</td>
</tr>
<tr>
<td>Type</td>
<td>Type: ControlType (optional)</td>
<td>String indicating the control type. Inherited from ControlMetadata.Type</td>
</tr>
</tbody>
</table>
Field module

11/24/2021 • 4 minutes to read • Edit Online

Represents the run-time instance of a field.

Index

Types
- Field
- FieldDesign
- FieldMetadata

Types

Field

Hierarchy
InputControl
  └ Field

Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>container: boolean (optional)</td>
<td>True if the control is a container. Inherited from Control.container</td>
</tr>
<tr>
<td>generic</td>
<td>generic: boolean (optional)</td>
<td>Inherited from Control.generic</td>
</tr>
<tr>
<td>getDataSource</td>
<td>getDataSource: function(): any</td>
<td>Inherited from Control.getDataSource</td>
</tr>
<tr>
<td>hidden</td>
<td>hidden: boolean</td>
<td>True if the control is hidden. Inherited from Control.hidden</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>applyDesign</td>
<td>applyDesign(IDesign: FieldDesign): void</td>
<td>Applies given design to the design on the control. Overrides Control.applyDesign</td>
</tr>
<tr>
<td>dataContext</td>
<td>dataContext(): any</td>
<td>Inherited from Control.dataContext</td>
</tr>
<tr>
<td>getDesign</td>
<td>getDesign(): Design</td>
<td>Returns the design object of this control. Inherited from Control.getDesign</td>
</tr>
<tr>
<td>getEditableFormattedValue</td>
<td>getEditableFormattedValue(): string</td>
<td>Gets a formatted decimal string value of an editable field control.</td>
</tr>
</tbody>
</table>

### Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>getEditableValue</td>
<td>getEditableValue(): string</td>
<td>Gets the value for an editable field control.</td>
</tr>
<tr>
<td>getEntityRef</td>
<td>getEntityRef(): any</td>
<td>Gets value of entityRef binding to control.</td>
</tr>
<tr>
<td>getFormattedValue</td>
<td>getFormattedValue(): string</td>
<td>Gets a formatted decimal string value.</td>
</tr>
<tr>
<td>getRefLink</td>
<td>getRefLink(): NavigationArgs</td>
<td>Gets the navigation object for a reference link.</td>
</tr>
<tr>
<td>getValue</td>
<td>getValue(): any</td>
<td>Gets the value for a field control.</td>
</tr>
<tr>
<td>hasRefLink</td>
<td>hasRefLink(): boolean</td>
<td>Returns true if the field has a refLink, otherwise false.</td>
</tr>
<tr>
<td>hasUnWrapText</td>
<td>hasUnWrapText(): boolean</td>
<td>Gets wrap text property of control.</td>
</tr>
<tr>
<td>isEditable</td>
<td>isEditable(): boolean</td>
<td>Boolean indicating if the control is editable.</td>
</tr>
<tr>
<td>metadata</td>
<td>metadata(): FieldMetadata</td>
<td>Returns the metadata object of this control.</td>
</tr>
<tr>
<td>parent</td>
<td>parent(): Control</td>
<td>Returns the parent (control or page) of this control.</td>
</tr>
<tr>
<td>root</td>
<td>root(): Page</td>
<td>Returns the root form instance (page) of this control.</td>
</tr>
<tr>
<td>setEditableValue</td>
<td>setEditableValue(value: string</td>
<td>Sets the value for an editable field control.</td>
</tr>
</tbody>
</table>

### Events

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>onDataChange</td>
<td>onDataChange: EventHook</td>
<td>An event that is triggered when the input control's data changes. Inherited from InputControl.onDataChanged</td>
</tr>
</tbody>
</table>

### FieldDesign

**Hierarchy**

- InputControlDesign
  - FieldDesign

**Properties**
<table>
<thead>
<tr>
<th><strong>NAME</strong></th>
<th><strong>SIGNATURE</strong></th>
<th><strong>DESCRIPTION</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>alignItems</td>
<td>alignItems: string (optional)</td>
<td>This property is an alias for the CSS property &quot;align-items&quot;. Inherited from Design.alignItems</td>
</tr>
<tr>
<td>alignSelf</td>
<td>alignSelf: string (optional)</td>
<td>Inherited from Design.alignSelf</td>
</tr>
<tr>
<td>bindings</td>
<td>bindings: any (optional)</td>
<td>Inherited from Design.bindings</td>
</tr>
<tr>
<td>border</td>
<td>border: &quot;none&quot;</td>
<td>The border behavior of a control. This property will not be inherited by the children. Inherited from Design.border</td>
</tr>
<tr>
<td>color</td>
<td>color: string (optional)</td>
<td>The foreground color of the container. Inherited from Design.color</td>
</tr>
<tr>
<td>flexFlow</td>
<td>flexFlow: string (optional)</td>
<td>Specifying this property makes the component a flex container component. Inherited from Design.flexFlow</td>
</tr>
<tr>
<td>flexSize</td>
<td>flexSize: string (optional)</td>
<td>One number or two numbers written as a string. For example, &quot;(size to grow) [(size-to-shrink)]&quot; to accommodate available space in the immediate flex container. Inherited from Design.flexSize</td>
</tr>
<tr>
<td>fontSize</td>
<td>fontSize: &quot;medium&quot;</td>
<td>The proportional text size Inherited from Design.fontSize</td>
</tr>
<tr>
<td>fontWeight</td>
<td>fontWeight: &quot;normal&quot;</td>
<td>Normal or bold text. Inherited from Design.fontWeight</td>
</tr>
<tr>
<td>justifyItems</td>
<td>justifyItems: &quot;flex-start&quot;</td>
<td>This property is an alias for the CSS property &quot;justify-content&quot;. Inherited from Design.justifyItems</td>
</tr>
<tr>
<td>label</td>
<td>label: string (optional)</td>
<td>Inherited from Design.label</td>
</tr>
<tr>
<td>labelPosition</td>
<td>labelPosition: &quot;stacked&quot;</td>
<td>Determines how a label is positioned, if at all. By default, labelPosition is set to stacked. Inherited from Design.labelPosition</td>
</tr>
<tr>
<td>name</td>
<td>name: string (optional)</td>
<td>Inherited from Design.name</td>
</tr>
<tr>
<td>padding</td>
<td>padding: &quot;none&quot;</td>
<td>Allows specifying the component's padding behavior. Inherited from Design.padding</td>
</tr>
<tr>
<td>type</td>
<td>type: ControlType (optional)</td>
<td>The type of the control as a string. Inherited from Design.type</td>
</tr>
</tbody>
</table>
# FieldMetadata

## Hierarchy

```
  InputControlMetadata
      └ FieldMetadata
```

## Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoundEntity</td>
<td>BoundEntity: string (optional)</td>
<td>The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity</td>
</tr>
<tr>
<td>BoundField</td>
<td>BoundField: string (optional)</td>
<td>Inherited from ControlMetadata.BoundField</td>
</tr>
<tr>
<td>DecimalPlaces</td>
<td>DecimalPlaces: number (optional)</td>
<td>The number of decimals that appear on a field of type &quot;Real&quot;.</td>
</tr>
<tr>
<td>Description</td>
<td>Description: string (optional)</td>
<td>Description of the control. Inherited from ControlMetadata.Description</td>
</tr>
<tr>
<td>Editable</td>
<td>Editable: boolean (optional)</td>
<td>Boolean indicating if the control is editable. Inherited from ControlMetadata.Editable</td>
</tr>
<tr>
<td>ExtType</td>
<td>ExtType: ControlType (optional)</td>
<td>The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType</td>
</tr>
<tr>
<td>Formatting</td>
<td>Formatting: any (optional)</td>
<td>Formats a field of type &quot;DateTime&quot; or &quot;Date&quot;.</td>
</tr>
<tr>
<td>HelpText</td>
<td>HelpText: string (optional)</td>
<td>The keyboard shortcut for a command. For example, &quot;(Shift+F5)&quot;. For example, a control of type Input might have a label &quot;First Name&quot;. Inherited from ControlMetadata.HelpText</td>
</tr>
<tr>
<td>Hidden</td>
<td>Hidden: boolean (optional)</td>
<td>Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden</td>
</tr>
<tr>
<td>Id</td>
<td>Id: string (optional)</td>
<td>Identification string for a control. Inherited from ControlMetadata.Id</td>
</tr>
<tr>
<td>Label</td>
<td>Label: string (optional)</td>
<td>Label for a control. For example, a control representing a person's first name might have a label &quot;First Name&quot;. Inherited from ControlMetadata.Label</td>
</tr>
<tr>
<td><strong>NAME</strong></td>
<td><strong>SIGNATURE</strong></td>
<td><strong>DESCRIPTION</strong></td>
</tr>
<tr>
<td>----------</td>
<td>---------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>LinkType</td>
<td>LinkType: &quot;Telephone&quot;</td>
<td>Assigning the link type of a field allows for the appropriate mobile application to be opened when the link is selected.</td>
</tr>
<tr>
<td>Mandatory</td>
<td>Mandatory: boolean</td>
<td>If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline. Inherited from InputControlMetadata.Mandatory</td>
</tr>
<tr>
<td>Name</td>
<td>Name: string</td>
<td>Name of a control. Inherited from ControlMetadata.Name</td>
</tr>
<tr>
<td>NumSequence</td>
<td>NumSequence: NumberSequenceConfig</td>
<td>Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Inherited from InputControlMetadata.NumSequence</td>
</tr>
<tr>
<td>Order</td>
<td>Order: number</td>
<td>Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order</td>
</tr>
<tr>
<td>ReferenceAppId</td>
<td>ReferenceAppId: string</td>
<td>The ID of the app that the field control lives in.</td>
</tr>
<tr>
<td>ReferencePageId</td>
<td>ReferencePageId: string</td>
<td>The ID of the page that the field control lives in.</td>
</tr>
<tr>
<td>Style</td>
<td>Style: string</td>
<td>Styles a field of type &quot;DateTime&quot; or &quot;Date&quot;.</td>
</tr>
<tr>
<td>Type</td>
<td>Type: ControlType</td>
<td>String indicating the control type. Inherited from ControlMetadata.Type</td>
</tr>
<tr>
<td>UnWrapText</td>
<td>UnWrapText: boolean</td>
<td>False by default -- text of the page will be wrapped.</td>
</tr>
<tr>
<td>WrapText</td>
<td>WrapText: boolean</td>
<td>If true then the text of the field control will wrap to the next line.</td>
</tr>
</tbody>
</table>
A control for uploading images.

Index

Types
- FileUploader
- FileUploaderDesign
- FileUploaderMetadata

Types

FileUploader

Hierarchy
Value
- FileUploader

Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>container: boolean (optional)</td>
<td>True if the control is a container. Inherited from Control.container</td>
</tr>
<tr>
<td>generic</td>
<td>generic: boolean (optional)</td>
<td>Inherited from Control.generic</td>
</tr>
<tr>
<td>getDataSource</td>
<td>getDataSource: function(): any</td>
<td>Inherited from Control.getDataSource</td>
</tr>
<tr>
<td>hidden</td>
<td>hidden: boolean</td>
<td>True if the control is hidden. Inherited from Control.hidden</td>
</tr>
<tr>
<td>image</td>
<td>image: Image</td>
<td></td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>applyDesign</td>
<td>applyDesign(IDesign: FileUploaderDesign): void</td>
<td>Applies given design to the design on the control. Overrides Control.applyDesign</td>
</tr>
<tr>
<td>canLoadFromDevice</td>
<td>canLoadFromDevice(): boolean</td>
<td>Returns true if the mobile phone has camera plugin.</td>
</tr>
<tr>
<td>dataContext</td>
<td>dataContext(): any</td>
<td>Inherited from Control.dataContext</td>
</tr>
<tr>
<td>getDesign</td>
<td>getDesign(): Design</td>
<td>Returns the design object of this control. Inherited from Control.getDesign</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>getImage</td>
<td>getImage(options: any): Promise &lt;string&gt;</td>
<td>Returns a promise of an object with image data.</td>
</tr>
<tr>
<td>getValue</td>
<td>getValue(): any</td>
<td>Gets the value of the entity that is bound to the control. Overrides Value.getValue</td>
</tr>
<tr>
<td>isEditable</td>
<td>isEditable(): boolean</td>
<td>Boolean indicating if the control is editable. Inherited from Control.isEditable</td>
</tr>
<tr>
<td>loadFromFileSystem</td>
<td>loadFromFileSystem(file: Blob): Promise &lt;any&gt;</td>
<td></td>
</tr>
<tr>
<td>metadata</td>
<td>metadata(): FileUploaderMetadata</td>
<td>Returns the metadata object of this control. Inherited from Value.metadata</td>
</tr>
<tr>
<td>parent</td>
<td>parent(): Control</td>
<td>Page</td>
</tr>
<tr>
<td>root</td>
<td>root(): Page</td>
<td>Returns the root form instance (page) of this control. Inherited from Control.root</td>
</tr>
<tr>
<td>setCamera</td>
<td>setCamera(camera: any): void</td>
<td>Set the camera object on the control.</td>
</tr>
<tr>
<td>setValue</td>
<td>setValue(value: string): void</td>
<td>Sets the value of the control. Inherited from Value.setValue</td>
</tr>
</tbody>
</table>

**Events**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>onDataChange</td>
<td>onDataChange: EventHook &lt;null&gt;</td>
<td>An event that is triggered when the input control's data changes. Inherited from InputControl.onDataChanged</td>
</tr>
</tbody>
</table>

**FileUploaderDesign**

**Hierarchy**

- ValueDesign
  - FileUploaderDesign

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>alignItems</td>
<td>alignItems: string (optional)</td>
<td>This property is an alias for the CSS property &quot;align-items&quot;. Inherited from Design.alignItems</td>
</tr>
<tr>
<td>alignSelf</td>
<td>alignSelf: string (optional)</td>
<td>Inherited from Design.alignSelf</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bindings</td>
<td>bindings: any (optional)</td>
<td>Inherited from Design.bindings</td>
</tr>
<tr>
<td>border</td>
<td>border: &quot;none&quot;</td>
<td>&quot;solid&quot;</td>
</tr>
<tr>
<td>color</td>
<td>color: string (optional)</td>
<td>The foreground color of the container. Inherited from Design.color</td>
</tr>
<tr>
<td>flexFlow</td>
<td>flexFlow: string (optional)</td>
<td>Specifying this property makes the component a flex container component. Inherited from Design.flexFlow</td>
</tr>
<tr>
<td>flexSize</td>
<td>flexSize: string (optional)</td>
<td>One number or two numbers written as a string. For example, &quot;(size to grow) [(size-to-shrink)]&quot; to accommodate available space in the immediate flex container. Inherited from Design.flexSize</td>
</tr>
<tr>
<td>fontSize</td>
<td>fontSize: &quot;medium&quot;</td>
<td>&quot;xx-small&quot;</td>
</tr>
<tr>
<td>fontWeight</td>
<td>fontWeight: &quot;normal&quot;</td>
<td>&quot;bold&quot; (optional)</td>
</tr>
<tr>
<td>justifyItems</td>
<td>justifyItems: &quot;flex-start&quot;</td>
<td>&quot;flex-end&quot;</td>
</tr>
<tr>
<td>label</td>
<td>label: string (optional)</td>
<td>Inherited from Design.label</td>
</tr>
<tr>
<td>labelPosition</td>
<td>labelPosition: &quot;stacked&quot;</td>
<td>&quot;hidden&quot;</td>
</tr>
<tr>
<td>name</td>
<td>name: string (optional)</td>
<td>Inherited from Design.name</td>
</tr>
<tr>
<td>padding</td>
<td>padding: &quot;none&quot;</td>
<td>&quot;small&quot;</td>
</tr>
<tr>
<td>type</td>
<td>type: ControlType (optional)</td>
<td>The type of the control as a string. Inherited from Design.type</td>
</tr>
</tbody>
</table>

**FileUploaderMetadata**

**Hierarchy**

- ValueMetadata
  - FileUploaderMetadata

**Properties**
<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoundEntity</td>
<td>BoundEntity: string (optional)</td>
<td>The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity</td>
</tr>
<tr>
<td>BoundField</td>
<td>BoundField: string (optional)</td>
<td>Inherited from ControlMetadata.BoundField</td>
</tr>
<tr>
<td>Description</td>
<td>Description: string (optional)</td>
<td>Description of the control. Inherited from ControlMetadata.Description</td>
</tr>
<tr>
<td>Editable</td>
<td>Editable: boolean (optional)</td>
<td>Boolean indicating if the control is editable. Inherited from ControlMetadata.Editable</td>
</tr>
<tr>
<td>ExtType</td>
<td>ExtType: ControlType (optional)</td>
<td>The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType</td>
</tr>
<tr>
<td>HelpText</td>
<td>HelpText: string (optional)</td>
<td>The keyboard shortcut for a command. For example, &quot;(Shift+F5)&quot;. Inherited from ControlMetadata.HelpText</td>
</tr>
<tr>
<td>Hidden</td>
<td>Hidden: boolean (optional)</td>
<td>Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden</td>
</tr>
<tr>
<td>Id</td>
<td>Id: string (optional)</td>
<td>Identification string for a control. Inherited from ControlMetadata.Id</td>
</tr>
<tr>
<td>Label</td>
<td>Label: string (optional)</td>
<td>Label for a control. For example, a control representing a person's first name might have a label &quot;First Name&quot;. Inherited from ControlMetadata.Label</td>
</tr>
<tr>
<td>Mandatory</td>
<td>Mandatory: boolean (optional)</td>
<td>If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline. Inherited from InputControlMetadata.Mandatory</td>
</tr>
<tr>
<td>Name</td>
<td>Name: string (optional)</td>
<td>Name of a control. Inherited from ControlMetadata.Name</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NumSequence</td>
<td>NumSequence: NumberSequenceConfig (optional)</td>
<td>Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Inherited from InputControlMetadata.NumSequence</td>
</tr>
<tr>
<td>Order</td>
<td>Order: number (optional)</td>
<td>Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order</td>
</tr>
<tr>
<td>Type</td>
<td>Type: ControlType (optional)</td>
<td>String indicating the control type. Inherited from ControlMetadata.Type</td>
</tr>
</tbody>
</table>
A group control is a container control that has any number of controls as children.

**Index**

**Types**
- Group
- GroupDesign
- GroupMetadata

**Types**

**Group**

**Hierarchy**
ContainerControl
  └ Group

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>container: boolean</td>
<td>True if the control is a container. Inherited from ContainerControl.container</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Overrides Control.container</td>
</tr>
<tr>
<td>generic</td>
<td>generic: boolean (optional)</td>
<td>Inherited from Control.generic</td>
</tr>
<tr>
<td>getDataSource</td>
<td>getDataSource: function(): any</td>
<td>Inherited from Control.getDataSource</td>
</tr>
<tr>
<td>hidden</td>
<td>hidden: boolean</td>
<td>True if the control is hidden. Inherited from Control.hidden</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>applyDesign</td>
<td>applyDesign(!Design: GroupDesign): void</td>
<td>Applies given design to the design on the control. Overrides Control.applyDesign</td>
</tr>
<tr>
<td>dataContext</td>
<td>dataSource(): any</td>
<td>Inherited from Control.dataContext</td>
</tr>
<tr>
<td>getChildren</td>
<td>getChildren(): Control []</td>
<td>Returns the list of children associated with this group control.</td>
</tr>
<tr>
<td>getControl</td>
<td>getControl(controlName: string): Control</td>
<td>Given the name of a control, returns the control instance. Inherited from ContainerControl.getControl</td>
</tr>
</tbody>
</table>
### getControlById

**getControlById(id: string): Control**

Given the ID of a control, returns the control instance. Inherited from `ContainerControl.getControlById`.

### getDesign

**getDesign(): Design**

Returns the design object of this control. Inherited from `Control.getDesign`.

### isEditable

**isEditable(): boolean**

Boolean indicating if the control is editable. Inherited from `Control.isEditable`.

### metadata

**metadata(): GroupMetadata**

Returns the metadata object of this control. Overrides `ContainerControl.metadata`.

### parent

**parent(): Control | Page**

Returns the parent (control or page) of this control. Inherited from `Control.parent`.

### root

**root(): Page**

Returns the root form instance (page) of this control. Inherited from `Control.root`.

---

**GroupDesign**

**Hierarchy**

- **ContainerControlDesign**
  - **GroupDesign**

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>alignItems</td>
<td>alignItems: string (optional)</td>
<td>This property is an alias for the CSS property &quot;align-items&quot;. Inherited from <code>Design.alignItems</code>.</td>
</tr>
<tr>
<td>alignSelf</td>
<td>alignSelf: string (optional)</td>
<td>Inherited from <code>Design.alignSelf</code>.</td>
</tr>
<tr>
<td>allowScroll</td>
<td>allowScroll: string (optional)</td>
<td>True if the container will allow scrolling when its items do not fit into the container’s available space. If a container has an item which may scroll, then set this property to false to prevent nested scrolling areas. Inherited from <code>ContainerControlDesign.allowScroll</code></td>
</tr>
<tr>
<td>background</td>
<td>background: string (optional)</td>
<td>The background color of the container. Inherited from <code>ContainerControlDesign.background</code></td>
</tr>
<tr>
<td>bindings</td>
<td>bindings: any (optional)</td>
<td>Inherited from <code>Design.bindings</code></td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>border</td>
<td>border: &quot;none&quot;</td>
<td>&quot;solid&quot;</td>
</tr>
<tr>
<td>color</td>
<td>color: string (optional)</td>
<td>The foreground color of the container. Inherited from Design.color</td>
</tr>
<tr>
<td>flexFlow</td>
<td>flexFlow: string (optional)</td>
<td>Specifying this property makes the component a flex container component. Inherited from Design.flexFlow</td>
</tr>
<tr>
<td>flexSize</td>
<td>flexSize: string (optional)</td>
<td>One number or two numbers written as a string. For example, &quot;(size to grow) [(size-to-shrink)]&quot; to accommodate available space in the immediate flex container. Inherited from Design.flexSize</td>
</tr>
<tr>
<td>fontSize</td>
<td>fontSize: &quot;medium&quot;</td>
<td>&quot;xx-small&quot;</td>
</tr>
<tr>
<td>fontWeight</td>
<td>fontWeight: &quot;normal&quot;</td>
<td>&quot;bold&quot; (optional)</td>
</tr>
<tr>
<td>itemBorder</td>
<td>itemBorder: &quot;solid&quot;</td>
<td>&quot;none&quot; (optional)</td>
</tr>
<tr>
<td>items</td>
<td>items: string</td>
<td>Design [ ] (optional)</td>
</tr>
<tr>
<td>justifyItems</td>
<td>justifyItems: &quot;flex-start&quot;</td>
<td>&quot;flex-end&quot;</td>
</tr>
<tr>
<td>label</td>
<td>label: string (optional)</td>
<td>Inherited from Design.label</td>
</tr>
<tr>
<td>labelPosition</td>
<td>labelPosition: &quot;stacked&quot;</td>
<td>&quot;hidden&quot;</td>
</tr>
<tr>
<td>name</td>
<td>name: string (optional)</td>
<td>Inherited from Design.name</td>
</tr>
<tr>
<td>padding</td>
<td>padding: &quot;none&quot;</td>
<td>&quot;small&quot;</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------</td>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>type</td>
<td>type: ControlType (optional)</td>
<td>The type of the control as a string. Inherited from Design.type</td>
</tr>
</tbody>
</table>

**GroupMetadata**

**Hierarchy**

- ContainerControlMetadata
  - GroupMetadata

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoundEntity</td>
<td>BoundEntity: string (optional)</td>
<td>The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity</td>
</tr>
<tr>
<td>BoundField</td>
<td>BoundField: string (optional)</td>
<td>Inherited from ControlMetadata.BoundField</td>
</tr>
<tr>
<td>Children</td>
<td>Children: ControlMetadata [] (optional)</td>
<td>List of control metadata for each child control.</td>
</tr>
<tr>
<td>Description</td>
<td>Description: string (optional)</td>
<td>Description of the control. Inherited from ControlMetadata.Description</td>
</tr>
<tr>
<td>Editable</td>
<td>Editable: boolean (optional)</td>
<td>Boolean indicating if the control is editable. Inherited from ControlMetadataEditable</td>
</tr>
<tr>
<td>ExtType</td>
<td>ExtType: ControlType (optional)</td>
<td>The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType</td>
</tr>
<tr>
<td>HelpText</td>
<td>HelpText: string (optional)</td>
<td>The keyboard shortcut for a command. For example, &quot;(Shift + F5)&quot;. Inherited from ControlMetadata.HelpText</td>
</tr>
<tr>
<td>Hidden</td>
<td>Hidden: boolean (optional)</td>
<td>Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden</td>
</tr>
<tr>
<td>Id</td>
<td>Id: string (optional)</td>
<td>Identification string for a control. Inherited from ControlMetadata.Id</td>
</tr>
<tr>
<td>Label</td>
<td>Label: string (optional)</td>
<td>Label for a control. For example, a control representing a person's first name might have a label &quot;First Name&quot;. Inherited from ControlMetadata.Label</td>
</tr>
<tr>
<td>Name</td>
<td>Signature</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Name</td>
<td>Name: string (optional)</td>
<td>Name of a control. Inherited from ControlMetadata.Name</td>
</tr>
<tr>
<td>Order</td>
<td>Order: number (optional)</td>
<td>Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order</td>
</tr>
<tr>
<td>Type</td>
<td>Type: ControlType (optional)</td>
<td>String indicating the control type. Inherited from ControlMetadata.Type</td>
</tr>
</tbody>
</table>
Hyperlink control is a control to represent hyperlinks. Pagelinks can also be used in most cases.

Index

Types
- HyperLink
- HyperLinkDesign
- HyperLinkMetadata

Types

HyperLink

Hierarchy
Value
  └ HyperLink

Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>container: boolean (optional)</td>
<td>True if the control is a container. Inherited from Control.container</td>
</tr>
<tr>
<td>generic</td>
<td>generic: boolean (optional)</td>
<td>Inherited from Control.generic</td>
</tr>
<tr>
<td>dataSource</td>
<td>dataSource: function(): any</td>
<td>Inherited from Control.dataSource</td>
</tr>
<tr>
<td>hidden</td>
<td>hidden: boolean</td>
<td>True if the control is hidden. Inherited from Control.hidden</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>applyDesign</td>
<td>applyDesign(IDesign: HyperLinkDesign): void</td>
<td>Applies given design to the design on the control. Overrides Control.applyDesign</td>
</tr>
<tr>
<td>dataContext</td>
<td>dataContext(): any</td>
<td>Inherited from Control.dataContext</td>
</tr>
<tr>
<td>getDesign</td>
<td>getDesign(): Design</td>
<td>Returns the design object of this control. Inherited from Control.getDesign</td>
</tr>
<tr>
<td>getHyperLinkValue</td>
<td>getHyperLinkValue(): string</td>
<td>Returns the value of the control. Inherited from Value.getHyperLinkValue</td>
</tr>
<tr>
<td>getValue</td>
<td>getValue(): string</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------</td>
</tr>
<tr>
<td>isEditable</td>
<td>isEditable(): boolean</td>
<td>Boolean indicating if the control is editable. Inherited from Control.isEditable</td>
</tr>
<tr>
<td>isHyperLinkURLPresent</td>
<td>isHyperLinkURLPresent(): boolean</td>
<td></td>
</tr>
<tr>
<td>metadata</td>
<td>metadata(): HyperLinkMetadata</td>
<td>Returns the metadata object of this control. Overrides Value.metadata</td>
</tr>
<tr>
<td>parent</td>
<td>parent(): Control</td>
<td>Page Returns the parent (control or page) of this control. Inherited from Control.parent</td>
</tr>
<tr>
<td>root</td>
<td>root(): Page</td>
<td>Returns the root form instance (page) of this control. Inherited from Control.root</td>
</tr>
<tr>
<td>setBaseURL</td>
<td>setBaseURL(url: string): any</td>
<td></td>
</tr>
<tr>
<td>setValue</td>
<td>setValue(value: string): void</td>
<td>Sets the value of the control. Inherited from Value.setValue</td>
</tr>
</tbody>
</table>

**Events**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>onDataChange</td>
<td>onDataChange: EventHook &lt;null&gt;</td>
<td>An event that is triggered when the input control's data changes. Inherited from INPUTControl.onDataChanged</td>
</tr>
</tbody>
</table>

**HyperLinkDesign**

**Hierarchy**

ValueDesign

└—— HyperLinkDesign

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>alignItems</td>
<td>alignItems: string (optional)</td>
<td>This property is an alias for the CSS property &quot;align-items&quot;. Inherited from Design.alignItems</td>
</tr>
<tr>
<td>alignSelf</td>
<td>alignSelf: string (optional)</td>
<td>Inherited from Design.alignSelf</td>
</tr>
<tr>
<td>bindings</td>
<td>bindings: any (optional)</td>
<td>Inherited from Design.bindings</td>
</tr>
<tr>
<td>border</td>
<td>border: &quot;none&quot;</td>
<td>&quot;solid&quot;</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------------</td>
<td>----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>color</td>
<td>color: string (optional)</td>
<td>The foreground color of the container. Inherited from Design.color</td>
</tr>
<tr>
<td>flexFlow</td>
<td>flexFlow: string (optional)</td>
<td>Specifying this property makes the component a flex container component. Inherited from Design.flexFlow</td>
</tr>
<tr>
<td>flexSize</td>
<td>flexSize: string (optional)</td>
<td>One number or two numbers written as a string. For example, &quot;(size to grow) [(size-to-shrink)]&quot; to accommodate available space in the immediate flex container. Inherited from Design.flexSize</td>
</tr>
<tr>
<td>fontSize</td>
<td>fontSize: &quot;medium&quot;</td>
<td>&quot;xx-small&quot;</td>
</tr>
<tr>
<td>fontWeight</td>
<td>fontWeight: &quot;normal&quot;</td>
<td>&quot;bold&quot; (optional)</td>
</tr>
<tr>
<td>justifyItems</td>
<td>justifyItems: &quot;flex-start&quot;</td>
<td>&quot;flex-end&quot;</td>
</tr>
<tr>
<td>label</td>
<td>label: string (optional)</td>
<td>Inherited from Design.label</td>
</tr>
<tr>
<td>labelPosition</td>
<td>labelPosition: &quot;stacked&quot;</td>
<td>&quot;hidden&quot;</td>
</tr>
<tr>
<td>name</td>
<td>name: string (optional)</td>
<td>Inherited from Design.name</td>
</tr>
<tr>
<td>padding</td>
<td>padding: &quot;none&quot;</td>
<td>&quot;small&quot;</td>
</tr>
<tr>
<td>type</td>
<td>type: ControlType (optional)</td>
<td>The type of the control as a string. Inherited from Design.type</td>
</tr>
</tbody>
</table>

HyperLinkMetadata

Hierarchy

ValueMetadata

---

BoundEntity: string (optional) The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity
<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoundField</td>
<td>BoundField: string (optional)</td>
<td>Inherited from ControlMetadata.BoundField</td>
</tr>
<tr>
<td>Description</td>
<td>Description: string (optional)</td>
<td>Description of the control. Inherited from ControlMetadata.Description</td>
</tr>
<tr>
<td>Editable</td>
<td>Editable: boolean (optional)</td>
<td>Boolean indicating if the control is editable. Inherited from ControlMetadataEditable</td>
</tr>
<tr>
<td>ExtType</td>
<td>ExtType: ControlType (optional)</td>
<td>The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType</td>
</tr>
<tr>
<td>HelpText</td>
<td>HelpText: string (optional)</td>
<td>The keyboard shortcut for a command. For example, &quot;(Shift+F5)&quot; Inherited from ControlMetadata.HelpText</td>
</tr>
<tr>
<td>Hidden</td>
<td>Hidden: boolean (optional)</td>
<td>Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden</td>
</tr>
<tr>
<td>Id</td>
<td>Id: string (optional)</td>
<td>Identification string for a control. Inherited from ControlMetadata.Id</td>
</tr>
<tr>
<td>Label</td>
<td>Label: string (optional)</td>
<td>Label for a control. For example, a control representing a person's first name might have a label &quot;First Name&quot;. Inherited from ControlMetadata.Label</td>
</tr>
<tr>
<td>Mandatory</td>
<td>Mandatory: boolean (optional)</td>
<td>If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline. Inherited from InputControlMetadata.Mandatory</td>
</tr>
<tr>
<td>Name</td>
<td>Name: string (optional)</td>
<td>Name of a control. Inherited from ControlMetadata.Name</td>
</tr>
<tr>
<td>NumSequence</td>
<td>NumSequence: NumberSequenceConfig (optional)</td>
<td>Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Inherited from InputControlMetadata.NumSequence</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Order</td>
<td>Order: number (optional)</td>
<td>Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order</td>
</tr>
<tr>
<td>Type</td>
<td>Type: ControlType (optional)</td>
<td>String indicating the control type. Inherited from ControlMetadata.Type</td>
</tr>
</tbody>
</table>
Image control for representing images in the mobile app. Images can be of any of the following types: DataUri, Base64, URL, AOTResource, or Symbol.

Index

Types
- Image
- ImageDesign
- ImageMetadata

Type aliases
- ImageStyleType

Types

Image

Hierarchy
Control
- Image

Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>container: boolean (optional)</td>
<td>True if the control is a container. Inherited from Control.container</td>
</tr>
<tr>
<td>generic</td>
<td>generic: boolean (optional)</td>
<td>Inherited from Control.generic</td>
</tr>
<tr>
<td>getDataSource</td>
<td>getDataSource: function(): any</td>
<td>Inherited from Control.getDataSource</td>
</tr>
<tr>
<td>hidden</td>
<td>hidden: boolean</td>
<td>True if the control is hidden. Inherited from Control.hidden</td>
</tr>
<tr>
<td>imageSource</td>
<td>imageSource: string</td>
<td>Defines the imageSource.</td>
</tr>
<tr>
<td>imageView</td>
<td>imageView: string</td>
<td>Dictates the style of the image.</td>
</tr>
<tr>
<td>placeholderClass</td>
<td>placeholderClass: string</td>
<td></td>
</tr>
<tr>
<td>symbol</td>
<td>symbol: string</td>
<td>Defines the symbol if the image is of type symbol.</td>
</tr>
</tbody>
</table>

Methods
<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>applyDesign</td>
<td>applyDesign(design: ImageDesign): void</td>
<td>Applies given design to the design on the control. Overrides Control.applyDesign</td>
</tr>
<tr>
<td>dataContext</td>
<td>dataContext(): any</td>
<td>Inherited from Control.dataContext</td>
</tr>
<tr>
<td>getDesign</td>
<td>getDesign(): Design</td>
<td>Returns the design object of this control. Inherited from Control.getDesign</td>
</tr>
<tr>
<td>isEditable</td>
<td>isEditable(): boolean</td>
<td>Boolean indicating if the control is editable. Inherited from Control.isEditable</td>
</tr>
<tr>
<td>metadata</td>
<td>metadata(): ImageMetadata</td>
<td>Returns the metadata object of this control. Overrides Control.metadata</td>
</tr>
<tr>
<td>parent</td>
<td>parent(): Control</td>
<td>Page</td>
</tr>
<tr>
<td>root</td>
<td>root(): Page</td>
<td>Returns the root form instance (page) of this control. Inherited from Control.root</td>
</tr>
</tbody>
</table>

**ImageDesign**

Hierarchy

Design

└—— ImageDesign

Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>alignItems</td>
<td>alignItems: string (optional)</td>
<td>This property is an alias for the CSS property &quot;align-items&quot;. Inherited from Design.alignItems</td>
</tr>
<tr>
<td>alignSelf</td>
<td>alignSelf: string (optional)</td>
<td>Inherited from Design.alignSelf</td>
</tr>
<tr>
<td>bindings</td>
<td>bindings: any (optional)</td>
<td>Inherited from Design.bindings</td>
</tr>
<tr>
<td>border</td>
<td>border: &quot;none&quot;</td>
<td>&quot;solid&quot;</td>
</tr>
<tr>
<td>color</td>
<td>color: string (optional)</td>
<td>The foreground color of the container. Inherited from Design.color</td>
</tr>
<tr>
<td>flexFlow</td>
<td>flexFlow: string (optional)</td>
<td>Specifying this property makes the component a flex container component. Inherited from Design.flexFlow</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>flexSize</td>
<td>flexSize: string (optional)</td>
<td>One number or two numbers written as a string. For example, &quot;(size to grow) [(size-to-shrink)]&quot; to accommodate available space in the immediate flex container. Inherited from Design.flexSize</td>
</tr>
<tr>
<td>fontSize</td>
<td>fontSize: &quot;medium&quot;</td>
<td>The proportional text size Inherited from Design.fontSize</td>
</tr>
<tr>
<td>fontWeight</td>
<td>fontWeight: &quot;normal&quot;</td>
<td>Normal or bold text. Inherited from Design.fontWeight</td>
</tr>
<tr>
<td>height</td>
<td>height: string (optional)</td>
<td>The relative vertical size of the image.</td>
</tr>
<tr>
<td>imageStyle</td>
<td>imageStyle: ImageStyleType (optional)</td>
<td>The style of the image.</td>
</tr>
<tr>
<td>justifyItems</td>
<td>justifyItems: &quot;flex-start&quot;</td>
<td>This property is an alias for the CSS property &quot;justify-content&quot;. Inherited from Design.justifyItems</td>
</tr>
<tr>
<td>label</td>
<td>label: string (optional)</td>
<td>Inherited from Design.label</td>
</tr>
<tr>
<td>labelPosition</td>
<td>labelPosition: &quot;stacked&quot;</td>
<td>Determines how a label is positioned, if at all. By default, labelPosition is set to stacked. Inherited from Design.labelPosition</td>
</tr>
<tr>
<td>name</td>
<td>name: string (optional)</td>
<td>Inherited from Design.name</td>
</tr>
<tr>
<td>padding</td>
<td>padding: &quot;none&quot;</td>
<td>Allows specifying the component’s padding behavior. Inherited from Design.padding</td>
</tr>
<tr>
<td>type</td>
<td>type: ControlType (optional)</td>
<td>The type of the control as a string. Inherited from Design.type</td>
</tr>
<tr>
<td>width</td>
<td>width: string (optional)</td>
<td>The relative horizontal size of the image.</td>
</tr>
</tbody>
</table>

**ImageMetadata**

**Hierarchy**

- ImageMetadata

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseUrl</td>
<td>BaseUrl: string (optional)</td>
<td>Base URL for AOTResource type image.</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BoundEntity</td>
<td>BoundEntity: string (optional)</td>
<td>The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity</td>
</tr>
<tr>
<td>BoundField</td>
<td>BoundField: string (optional)</td>
<td>Inherited from ControlMetadata.BoundField</td>
</tr>
<tr>
<td>Description</td>
<td>Description: string (optional)</td>
<td>Description of the control. Inherited from ControlMetadata.Description</td>
</tr>
<tr>
<td>Editable</td>
<td>Editable: boolean (optional)</td>
<td>Boolean indicating if the control is editable. Inherited from ControlMetadata.Editable</td>
</tr>
<tr>
<td>ExtType</td>
<td>ExtType: ControlType (optional)</td>
<td>The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType</td>
</tr>
<tr>
<td>Height</td>
<td>Height: number (optional)</td>
<td>The relative vertical size of the image.</td>
</tr>
<tr>
<td>HelpText</td>
<td>HelpText: string (optional)</td>
<td>The keyboard shortcut for a command. For example, &quot;(Shift+F5)&quot;. Inherited from ControlMetadata.HelpText</td>
</tr>
<tr>
<td>Hidden</td>
<td>Hidden: boolean (optional)</td>
<td>Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden</td>
</tr>
<tr>
<td>Id</td>
<td>Id: string (optional)</td>
<td>Identification string for a control. Inherited from ControlMetadata.Id</td>
</tr>
<tr>
<td>ImageStyle</td>
<td>ImageStyle: ImageStyleType (optional)</td>
<td>The style of the image.</td>
</tr>
<tr>
<td>Label</td>
<td>Label: string (optional)</td>
<td>Label for a control. For example, a control representing a person's first name might have a label &quot;First Name&quot;. Inherited from ControlMetadata.Label</td>
</tr>
<tr>
<td>Name</td>
<td>Name: string (optional)</td>
<td>Name of a control. Inherited from ControlMetadata.Name</td>
</tr>
<tr>
<td>Order</td>
<td>Order: number (optional)</td>
<td>Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order</td>
</tr>
<tr>
<td>Type</td>
<td>Type: ControlType (optional)</td>
<td>String indicating the control type. Inherited from ControlMetadata.Type</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Width</td>
<td>Width: number (optional)</td>
<td>The relative horizontal size of the image.</td>
</tr>
</tbody>
</table>

**Type aliases**

`ImageStyleType`

`ImageStyleType: "square" | "symbol" | "wide" | "circular" | "zoomable"`
Input controls are typically used on task pages for collecting user input, for example, for a new control.

Index

Types

- InputControl
- InputControlDesign
- InputControlMetadata
- NumberSequenceConfig

Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>container: boolean (optional)</td>
<td>True if the control is a container. Inherited from Control.container</td>
</tr>
<tr>
<td>generic</td>
<td>generic: boolean (optional)</td>
<td>Inherited from Control.generic</td>
</tr>
<tr>
<td>getDataSource</td>
<td>getDataSource: function(): any</td>
<td>Inherited from Control.getDataSource</td>
</tr>
<tr>
<td>hidden</td>
<td>hidden: boolean</td>
<td>True if the control is hidden. Inherited from Control.hidden</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>applyDesign</td>
<td>applyDesign(Design): void</td>
<td>Applies given design to the design on the control. Inherited from Control.applyDesign</td>
</tr>
<tr>
<td>dataContext</td>
<td>dataContext(): any</td>
<td>Inherited from Control.dataContext</td>
</tr>
<tr>
<td>getDesign</td>
<td>getDesign(): Design</td>
<td>Returns the design object of this control. Inherited from Control.getDesign</td>
</tr>
</tbody>
</table>
### isEditable

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>isEditable</td>
<td>isEditable(): boolean</td>
<td>Boolean indicating if the control is editable. Inherited from Control.isEditable</td>
</tr>
</tbody>
</table>

### metadata

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>metadata</td>
<td>metadata(): InputControlMetadata</td>
<td>Returns the metadata object of this control. Overrides Control.metadata</td>
</tr>
</tbody>
</table>

### parent

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent</td>
<td>parent(): Control</td>
<td>Returns the parent (control or page) of this control. Inherited from Control.parent</td>
</tr>
</tbody>
</table>

### root

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>root(): Page</td>
<td>Returns the root form instance (page) of this control. Inherited from Control.root</td>
</tr>
</tbody>
</table>

### Events

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>onDataChange</td>
<td>onDataChange: EventHook &lt;null&gt;</td>
<td>An event that is triggered when the input control's data changes.</td>
</tr>
</tbody>
</table>

### InputControlDesign

#### Hierarchy

- Design
  - InputControlDesign
    - FieldDesign
    - LookupDesign
    - MultiLookupDesign
    - ValueDesign

#### Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>alignItems</td>
<td>alignItems: string (optional)</td>
<td>This property is an alias for the CSS property “align-items”. Inherited from Design.alignItems</td>
</tr>
<tr>
<td>alignSelf</td>
<td>alignSelf: string (optional)</td>
<td>Inherited from Design.alignSelf</td>
</tr>
<tr>
<td>bindings</td>
<td>bindings: any (optional)</td>
<td>Inherited from Design.bindings</td>
</tr>
<tr>
<td>border</td>
<td>border: “none”</td>
<td>The border behavior of a control. This property will not be inherited by the children. Inherited from Design.border</td>
</tr>
<tr>
<td>color</td>
<td>color: string (optional)</td>
<td>The foreground color of the container. Inherited from Design.color</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>flexFlow</td>
<td>flexFlow: string (optional)</td>
<td>Specifying this property makes the component a flex container component. Inherited from Design.flexFlow</td>
</tr>
<tr>
<td>flexSize</td>
<td>flexSize: string (optional)</td>
<td>One number or two numbers written as a string. For example, &quot;(size to grow) [(size-to-shrink)]&quot; to accommodate available space in the immediate flex container. Inherited from Design.flexSize</td>
</tr>
<tr>
<td>fontSize</td>
<td>fontSize: &quot;medium&quot;</td>
<td>&quot;xx-small&quot;</td>
</tr>
<tr>
<td>fontWeight</td>
<td>fontWeight: &quot;normal&quot;</td>
<td>&quot;bold&quot; (optional)</td>
</tr>
<tr>
<td>justifyItems</td>
<td>justifyItems: &quot;flex-start&quot;</td>
<td>&quot;flex-end&quot;</td>
</tr>
<tr>
<td>label</td>
<td>label: string (optional)</td>
<td>Inherited from Design.label</td>
</tr>
<tr>
<td>labelPosition</td>
<td>labelPosition: &quot;stacked&quot;</td>
<td>&quot;hidden&quot;</td>
</tr>
<tr>
<td>name</td>
<td>name: string (optional)</td>
<td>Inherited from Design.name</td>
</tr>
<tr>
<td>padding</td>
<td>padding: &quot;none&quot;</td>
<td>&quot;small&quot;</td>
</tr>
<tr>
<td>type</td>
<td>type: ControlType (optional)</td>
<td>The type of the control as a string. Inherited from Design.type</td>
</tr>
</tbody>
</table>

**InputControlMetadata**

**Hierarchy**

ControlMetadata

- InputControlMetadata
  - FieldMetadata
  - LookupMetadata
  - MultiLookupMetadata
  - ValueMetadata

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>flexFlow</td>
<td>flexFlow: string (optional)</td>
<td>Specifying this property makes the component a flex container component. Inherited from Design.flexFlow</td>
</tr>
<tr>
<td>flexSize</td>
<td>flexSize: string (optional)</td>
<td>One number or two numbers written as a string. For example, &quot;(size to grow) [(size-to-shrink)]&quot; to accommodate available space in the immediate flex container. Inherited from Design.flexSize</td>
</tr>
<tr>
<td>fontSize</td>
<td>fontSize: &quot;medium&quot;</td>
<td>&quot;xx-small&quot;</td>
</tr>
<tr>
<td>fontWeight</td>
<td>fontWeight: &quot;normal&quot;</td>
<td>&quot;bold&quot; (optional)</td>
</tr>
<tr>
<td>justifyItems</td>
<td>justifyItems: &quot;flex-start&quot;</td>
<td>&quot;flex-end&quot;</td>
</tr>
<tr>
<td>label</td>
<td>label: string (optional)</td>
<td>Inherited from Design.label</td>
</tr>
<tr>
<td>labelPosition</td>
<td>labelPosition: &quot;stacked&quot;</td>
<td>&quot;hidden&quot;</td>
</tr>
<tr>
<td>name</td>
<td>name: string (optional)</td>
<td>Inherited from Design.name</td>
</tr>
<tr>
<td>padding</td>
<td>padding: &quot;none&quot;</td>
<td>&quot;small&quot;</td>
</tr>
<tr>
<td>type</td>
<td>type: ControlType (optional)</td>
<td>The type of the control as a string. Inherited from Design.type</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BoundEntity</td>
<td>BoundEntity: string (optional)</td>
<td>The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity</td>
</tr>
<tr>
<td>BoundField</td>
<td>BoundField: string (optional)</td>
<td>Inherited from ControlMetadata.BoundField</td>
</tr>
<tr>
<td>Description</td>
<td>Description: string (optional)</td>
<td>Description of the control. Inherited from ControlMetadata.Description</td>
</tr>
<tr>
<td>Editable</td>
<td>Editable: boolean (optional)</td>
<td>Boolean indicating if the control is editable. Inherited from ControlMetadata.Editable</td>
</tr>
<tr>
<td>ExtType</td>
<td>ExtType: ControlType (optional)</td>
<td>The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType</td>
</tr>
<tr>
<td>HelpText</td>
<td>HelpText: string (optional)</td>
<td>The keyboard shortcut for a command. For example, &quot;(Shift+F5)&quot; Inherited from ControlMetadata.HelpText</td>
</tr>
<tr>
<td>Hidden</td>
<td>Hidden: boolean (optional)</td>
<td>Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden</td>
</tr>
<tr>
<td>Id</td>
<td>Id: string (optional)</td>
<td>Identification string for a control. Inherited from ControlMetadata.Id</td>
</tr>
<tr>
<td>Label</td>
<td>Label: string (optional)</td>
<td>Label for a control. For example, a control representing a person's first name might have a label &quot;First Name&quot;. Inherited from ControlMetadata.Label</td>
</tr>
<tr>
<td>Mandatory</td>
<td>Mandatory: boolean (optional)</td>
<td>If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline.</td>
</tr>
<tr>
<td>Name</td>
<td>Name: string (optional)</td>
<td>Name of a control. Inherited from ControlMetadata.Name</td>
</tr>
<tr>
<td>NumSequence</td>
<td>NumSequence: NumberSequenceConfig (optional)</td>
<td>Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic.</td>
</tr>
</tbody>
</table>
**NumberSequenceConfig**

**Hierarchy**

NumberSequenceConfig

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order</td>
<td>Order: number (optional)</td>
<td>Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order</td>
</tr>
<tr>
<td>Type</td>
<td>Type: ControlType (optional)</td>
<td>String indicating the control type. Inherited from ControlMetadata.Type</td>
</tr>
</tbody>
</table>

**NumberSequenceConfig**

**Hierarchy**

NumberSequenceConfig

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataType</td>
<td>dataType: string</td>
<td>The data type is used to lookup whether the number sequence is editable or not on the reference page.</td>
</tr>
<tr>
<td>referencePageName</td>
<td>referencePageName: string</td>
<td>Page name of the page that defines if the num sequence is editable.</td>
</tr>
</tbody>
</table>
A list is a control that contains any numbers of rows. Each row follows a template for the layout of any number of controls. Lists come in two styles: simple and card.

Index

Types

- List
- ListDesign
- ListMetadata
- Row

Types

List

Hierarchy

ContainerControl

└ List

Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>$accessibility</td>
<td>$accessibility: any</td>
<td></td>
</tr>
<tr>
<td>DefaultSearchColumn</td>
<td>DefaultSearchColumn: string</td>
<td></td>
</tr>
<tr>
<td>container</td>
<td>container: boolean</td>
<td>True if the control is a container. Inherited from ContainerControl.container. Overrides Control.container.</td>
</tr>
<tr>
<td>emptyListMessage</td>
<td>emptyListMessage: string</td>
<td>Settable property to override default empty list message.</td>
</tr>
<tr>
<td>enableMultiSelect</td>
<td>enableMultiSelect: boolean</td>
<td></td>
</tr>
<tr>
<td>generic</td>
<td>generic: boolean (optional)</td>
<td>Inherited from Control.generic.</td>
</tr>
<tr>
<td>getDataSource</td>
<td>getDataSource: function(): any</td>
<td>Inherited from Control.getDataSource.</td>
</tr>
<tr>
<td>hidden</td>
<td>hidden: boolean</td>
<td>True if the control is hidden. Inherited from Control.hidden.</td>
</tr>
<tr>
<td>hideEmptyListMessage</td>
<td>hideEmptyListMessage: boolean</td>
<td>If true, no message is shown if the list is empty. To set this property, update the corresponding metadata property via configureControl.</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>imageFields</td>
<td>imageFields: any [ ]</td>
<td></td>
</tr>
<tr>
<td>performingRemoteSearch</td>
<td>performingRemoteSearch: boolean</td>
<td></td>
</tr>
<tr>
<td>searchQuery</td>
<td>searchQuery: [value: string]: any</td>
<td></td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>allowsNavigation</td>
<td>allowsNavigation(): boolean</td>
<td></td>
</tr>
<tr>
<td>applyDesign</td>
<td>applyDesign(IDesign: ListDesign): void</td>
<td>Applies given design to the design on the control. Overrides Control.applyDesign</td>
</tr>
<tr>
<td>applySearch</td>
<td>applySearch(): void</td>
<td></td>
</tr>
<tr>
<td>canPerformRemoteSearch</td>
<td>canPerformRemoteSearch(): boolean</td>
<td></td>
</tr>
<tr>
<td>clearSearch</td>
<td>clearSearch(): void</td>
<td></td>
</tr>
<tr>
<td>dataContext</td>
<td>dataContext(): any</td>
<td>Inherited from Control.dataContext</td>
</tr>
<tr>
<td>getColumnLabel</td>
<td>getColumnLabel(id: string): string</td>
<td></td>
</tr>
<tr>
<td>getControl</td>
<td>getControl(controlName: string): Control</td>
<td>Given the name of a control, returns the control instance. Inherited from ContainerControl.getControl</td>
</tr>
<tr>
<td>getControlById</td>
<td>getControlById(id: string): Control</td>
<td>Given the ID of a control, returns the control instance. Inherited from ContainerControl.getControlById</td>
</tr>
<tr>
<td>getControlMetadata</td>
<td>getControlMetadata(controlName: string): Control</td>
<td></td>
</tr>
<tr>
<td>getControlMetadataById</td>
<td>getControlMetadataById(id: string): Control</td>
<td></td>
</tr>
<tr>
<td>getData</td>
<td>getData(): any [ ]</td>
<td></td>
</tr>
<tr>
<td>getDesign</td>
<td>getDesign(): Design</td>
<td>Returns the design object of this control. Inherited from Control.getDesign</td>
</tr>
<tr>
<td>getListData</td>
<td>getListData(): any</td>
<td></td>
</tr>
<tr>
<td>getRenderedRows</td>
<td>getRenderedRows(): Row [ ]</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>getRowNavigation</code></td>
<td>`getRowNavigation(row: Row): Promise &lt;any&gt;</td>
<td>any`</td>
</tr>
<tr>
<td><code>getRowSelectionCount</code></td>
<td><code>getRowSelectionCount(): number</code></td>
<td></td>
</tr>
<tr>
<td><code>getRowSelections</code></td>
<td><code>getRowSelections(): string []</code></td>
<td></td>
</tr>
<tr>
<td><code>getRowTracking</code></td>
<td><code>getRowTracking(row: any, index: string): string</code></td>
<td></td>
</tr>
<tr>
<td><code>getSearchColumn</code></td>
<td><code>getSearchColumn(): string</code></td>
<td></td>
</tr>
<tr>
<td><code>getSearchColumnLabel</code></td>
<td><code>getSearchColumnLabel(): string</code></td>
<td></td>
</tr>
<tr>
<td><code>getSearchableColumns</code></td>
<td><code>getSearchableColumns(): any []</code></td>
<td></td>
</tr>
<tr>
<td><code>hideSearchBar</code></td>
<td><code>hideSearchBar(): boolean</code></td>
<td></td>
</tr>
<tr>
<td><code>isEditable</code></td>
<td><code>isEditable(): boolean</code></td>
<td>Boolean indicating if the control is editable. Inherited from <code>Control.isEditable</code></td>
</tr>
<tr>
<td><code>loadMetaData</code></td>
<td><code>loadMetaData(): void</code></td>
<td></td>
</tr>
<tr>
<td><code>loadMore</code></td>
<td><code>loadMore(): void</code></td>
<td></td>
</tr>
<tr>
<td><code>metadata</code></td>
<td><code>metadata(): ListMetadata</code></td>
<td>Returns the metadata object of this control. Overrides <code>ContainerControl.metadata</code></td>
</tr>
<tr>
<td><code>parent</code></td>
<td>`parent(): Control</td>
<td>Page`</td>
</tr>
<tr>
<td><code>performRemoteSearch</code></td>
<td><code>performRemoteSearch(): void</code></td>
<td></td>
</tr>
<tr>
<td><code>root</code></td>
<td><code>root(): Page</code></td>
<td>Returns the root form instance (page) of this control. Inherited from <code>Control.root</code></td>
</tr>
<tr>
<td><code>selectSearchColumn</code></td>
<td><code>selectSearchColumn(column: string): void</code></td>
<td></td>
</tr>
<tr>
<td><code>setRowSections</code></td>
<td><code>setRowSections(selections: string []): void</code></td>
<td></td>
</tr>
</tbody>
</table>

**Events**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>onRowCreate</td>
<td><code>onRowCreate: EventHook &lt;Row&gt;</code></td>
<td></td>
</tr>
<tr>
<td>onRowSelect</td>
<td><code>onRowSelect: EventHook &lt;Row&gt;</code></td>
<td></td>
</tr>
</tbody>
</table>
## ListDesign

**Hierarchy**

ContainerControlDesign

|-- ListDesign

### Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>alignItems</td>
<td>alignItems: string (optional)</td>
<td>This property is an alias for the CSS property &quot;align-items&quot;. Inherited from Design.alignItems</td>
</tr>
<tr>
<td>alignSelf</td>
<td>alignSelf: string (optional)</td>
<td>Inherited from Design.alignSelf</td>
</tr>
<tr>
<td>allowScroll</td>
<td>allowScroll: string (optional)</td>
<td>True if the container will allow scrolling when its items do not fit into the container's available space. If a container has an item which may scroll, then set this property to false to prevent nested scrolling areas. Inherited from ContainerControlDesign.allowScroll</td>
</tr>
<tr>
<td>background</td>
<td>background: string (optional)</td>
<td>The background color of the container. Inherited from ContainerControlDesign.background</td>
</tr>
<tr>
<td>bindings</td>
<td>bindings: any (optional)</td>
<td>Inherited from Design.bindings</td>
</tr>
<tr>
<td>border</td>
<td>border: &quot;none&quot;</td>
<td>&quot;solid&quot;</td>
</tr>
<tr>
<td>color</td>
<td>color: string (optional)</td>
<td>The foreground color of the container. Inherited from Design.color</td>
</tr>
<tr>
<td>design</td>
<td>design: GroupDesign (optional)</td>
<td>The design object that will be applied to each row.</td>
</tr>
<tr>
<td>flexFlow</td>
<td>flexFlow: string (optional)</td>
<td>Specifying this property makes the component a flex container component. Inherited from Design.flexFlow</td>
</tr>
<tr>
<td>flexSize</td>
<td>flexSize: string (optional)</td>
<td>One number or two numbers written as a string. For example, &quot;(size to grow) [(size-to-shrink)]&quot; to accommodate available space in the immediate flex container. Inherited from Design.flexSize</td>
</tr>
<tr>
<td>fontSize</td>
<td>fontSize: &quot;medium&quot;</td>
<td>&quot;xx-small&quot;</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>fontWeight</td>
<td>`fontWeight: &quot;normal&quot;</td>
<td>&quot;bold&quot; (optional)`</td>
</tr>
<tr>
<td>hideArrow</td>
<td><code>hideArrow: boolean (optional)</code></td>
<td>Allows an arrow ( &gt; ) on a default styled navigation control to be hidden.</td>
</tr>
<tr>
<td>hideSearchBar</td>
<td><code>hideSearchBar: boolean (optional)</code></td>
<td>If true, the search bar will be hidden.</td>
</tr>
<tr>
<td>itemBorder</td>
<td>`itemBorder: 'solid&quot;</td>
<td>&quot;none&quot; (optional)`</td>
</tr>
<tr>
<td>items</td>
<td>`items: string</td>
<td>Design [ ] (optional)`</td>
</tr>
<tr>
<td>justifyItems</td>
<td>`justifyItems: &quot;flex-start&quot;</td>
<td>&quot;flex-end&quot;</td>
</tr>
<tr>
<td>label</td>
<td><code>label: string (optional)</code></td>
<td>Inherited from Design.label</td>
</tr>
<tr>
<td>labelPosition</td>
<td>`labelPosition: &quot;stacked&quot;</td>
<td>&quot;hidden&quot;</td>
</tr>
<tr>
<td>name</td>
<td><code>name: string (optional)</code></td>
<td>Inherited from Design.name</td>
</tr>
<tr>
<td>padding</td>
<td>`padding: &quot;none&quot;</td>
<td>&quot;small&quot;</td>
</tr>
<tr>
<td>type</td>
<td><code>type: ControlType (optional)</code></td>
<td>The type of the control as a string. Inherited from Design.type</td>
</tr>
</tbody>
</table>

**ListMetadata**

**Hierarchy**

- ContainerControlMetadata
  - ListMetadata

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoundEntity</td>
<td><code>BoundEntity: string (optional)</code></td>
<td>The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity</td>
</tr>
<tr>
<td>BoundField</td>
<td><code>BoundField: string (optional)</code></td>
<td>Inherited from ControlMetadata.BoundField</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Children</td>
<td>Children: ControlMetadata [] (optional)</td>
<td>List of metadata for controls that will appear in each row of the list.</td>
</tr>
<tr>
<td>Description</td>
<td>Description: string (optional)</td>
<td>Description of the control. Inherited from ControlMetadata.Description</td>
</tr>
<tr>
<td>DetailsPageAppId</td>
<td>DetailsPageAppId: string (optional)</td>
<td>App ID of the page that each row in the list will navigate to.</td>
</tr>
<tr>
<td>DetailsPageld</td>
<td>DetailsPageld: string (optional)</td>
<td>The ID of the page to which each row will navigate.</td>
</tr>
<tr>
<td>Editable</td>
<td>Editable: boolean (optional)</td>
<td>Boolean indicating if the control is editable. Inherited from ControlMetadataEditable</td>
</tr>
<tr>
<td>EmptyListMessage</td>
<td>EmptyListMessage: string (optional)</td>
<td>If set, overrides the default message for empty lists.</td>
</tr>
<tr>
<td>ExtType</td>
<td>ExtType: ControlType (optional)</td>
<td>The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType</td>
</tr>
<tr>
<td>HelpText</td>
<td>HelpText: string (optional)</td>
<td>The keyboard shortcut for a command. For example, &quot;(Shift+F5)&quot; Inherited from ControlMetadata.HelpText</td>
</tr>
<tr>
<td>Hidden</td>
<td>Hidden: boolean (optional)</td>
<td>Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden</td>
</tr>
<tr>
<td>HideEmptyListMessage</td>
<td>HideEmptyListMessage: boolean (optional)</td>
<td>If true, the empty list message will be hidden.</td>
</tr>
<tr>
<td>HideSearchBar</td>
<td>HideSearchBar: boolean (optional)</td>
<td>If true, the search bar will be hidden.</td>
</tr>
<tr>
<td>Id</td>
<td>Id: string (optional)</td>
<td>Identification string for a control. Inherited from ControlMetadata.Id</td>
</tr>
<tr>
<td>InfiniteScroll</td>
<td>InfiniteScroll: boolean (optional)</td>
<td>If set to true then the list will allow infinite scroll.</td>
</tr>
<tr>
<td>InfiniteScrollPageSize</td>
<td>InfiniteScrollPageSize: number (optional)</td>
<td>Number of rows to load initially and the number of rows to load after the user reaches the end of the currently displayed rows.</td>
</tr>
<tr>
<td>Name</td>
<td>Signature</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Label</td>
<td>Label: string (optional)</td>
<td>Label for a control. For example, a control representing a person's first name might have a label “First Name”. Inherited from ControlMetadata.Label</td>
</tr>
<tr>
<td>ListStyle</td>
<td>ListStyle: string (optional)</td>
<td>Dictates the list template type.</td>
</tr>
<tr>
<td>MultiSelect</td>
<td>MultiSelect: boolean (optional)</td>
<td>If true, then the list will be a multi-select list.</td>
</tr>
<tr>
<td>Name</td>
<td>Name: string (optional)</td>
<td>Name of a control. Inherited from ControlMetadata.Name</td>
</tr>
<tr>
<td>NonEntityProjection</td>
<td>NonEntityProjection: boolean (optional)</td>
<td></td>
</tr>
<tr>
<td>Order</td>
<td>Order: number (optional)</td>
<td>Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order</td>
</tr>
<tr>
<td>Type</td>
<td>Type: ControlType (optional)</td>
<td>String indicating the control type. Inherited from ControlMetadata.Type</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>navigationHandler</td>
<td>Optional navigationHandler(row: Row): Promise &lt;any&gt;</td>
<td>A function that determines the navigation for a given row.</td>
</tr>
</tbody>
</table>

**Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnNavigate</td>
<td>OnNavigate: function(navigation: NavigationArgs): any (optional)</td>
<td>An event that is triggered when a pagelink control is selected.</td>
</tr>
<tr>
<td>OnRowSelect</td>
<td>OnRowSelect: function(row: Row): void (optional)</td>
<td>An event that is triggered when a row is selected.</td>
</tr>
</tbody>
</table>

**Row**

**Hierarchy**

**Row**

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldList</td>
<td>fieldList: Control []</td>
<td></td>
</tr>
<tr>
<td>headerField</td>
<td>headerField: Control</td>
<td></td>
</tr>
<tr>
<td>hidden</td>
<td>hidden: boolean</td>
<td>If true then the row will be hidden.</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>imageFields</td>
<td>imageFields: Control [ ]</td>
<td></td>
</tr>
<tr>
<td>isSelected</td>
<td>isSelected: boolean</td>
<td></td>
</tr>
<tr>
<td>item</td>
<td>item: any</td>
<td>A container of rendered data.</td>
</tr>
<tr>
<td>template</td>
<td>template: Group</td>
<td>Group control that represents the template for a row.</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>getControl</td>
<td>getControl(controlName: string): Control</td>
<td></td>
</tr>
<tr>
<td>getControlById</td>
<td>getControlById(id: string): Control</td>
<td></td>
</tr>
<tr>
<td>getControlValueById</td>
<td>getControlValueById(id: string): string</td>
<td></td>
</tr>
<tr>
<td>getRowHeader</td>
<td>getRowHeader(): Control</td>
<td></td>
</tr>
<tr>
<td>getRowId</td>
<td>getRowId(): string</td>
<td></td>
</tr>
<tr>
<td>hasImageField</td>
<td>hasImageField(): boolean</td>
<td>Returns true if the row has an image field.</td>
</tr>
<tr>
<td>isEntityCreatedNew</td>
<td>isEntityCreatedNew(): boolean</td>
<td></td>
</tr>
<tr>
<td>isEntityDeleted</td>
<td>isEntityDeleted(): boolean</td>
<td></td>
</tr>
<tr>
<td>isEntityModified</td>
<td>isEntityModified(): boolean</td>
<td></td>
</tr>
<tr>
<td>isEntitySyncPending</td>
<td>isEntitySyncPending(): boolean</td>
<td></td>
</tr>
<tr>
<td>select</td>
<td>select(): any</td>
<td></td>
</tr>
</tbody>
</table>
A lookup is an input control that is used to select an input from a list of options. For example, a lookup could be used to lookup a customer when linking a customer to a new sales order.

### Types

- **Lookup**
- **LookupDesign**
- **LookupMetadata**

### Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>container: boolean (optional)</td>
<td>True if the control is a container. Inherited from <code>Control.container</code></td>
</tr>
<tr>
<td>generic</td>
<td>generic: boolean (optional)</td>
<td>Inherited from <code>Control.generic</code></td>
</tr>
<tr>
<td>getDataSource</td>
<td>getDataSource: function(): any</td>
<td>Inherited from <code>Control.getDataSource</code></td>
</tr>
<tr>
<td>hidden</td>
<td>hidden: boolean</td>
<td>True if the control is hidden.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inherited from <code>Control.hidden</code></td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>applyDesign</td>
<td>applyDesign(IDesign: LookupDesign): void</td>
<td>Applies given design to the design on the control. Overrides <code>Control.applyDesign</code></td>
</tr>
<tr>
<td>dataContext</td>
<td>dataContext(): any</td>
<td>Inherited from <code>Control.dataContext</code></td>
</tr>
<tr>
<td>getDesign</td>
<td>getDesign(): Design</td>
<td>Returns the design object of this control.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inherited from <code>Control.getDesign</code></td>
</tr>
<tr>
<td>getDisplayValue</td>
<td>getDisplayValue(): string</td>
<td></td>
</tr>
<tr>
<td>getLookupPage</td>
<td>getLookupPage(): Page</td>
<td></td>
</tr>
</tbody>
</table>
### getValue

getValue(): string | number

**Description**

Boolean indicating if the control is editable. Inherited from Control.isEditable

### isEditable

isEditable(): boolean

**Description**

Boolean indicating if the control is editable. Inherited from Control.isEditable

### metadata

metadata(): LookupMetadata

**Description**

Returns the metadata object of this control. Overrides InputControl.metadata

### parent

parent(): Control | Page

**Description**

Returns the parent (control or page) of this control. Inherited from Control.parent

### root

root(): Page

**Description**

Returns the root form instance (page) of this control. Inherited from Control.root

### setEntityRef

setEntityRef(newValue: string | number): Promise <any>

### Events

**onDataChanged**

onDataChanged: EventHook <null>

**Description**

An event that is triggered when the input control's data changes. Inherited from InputControl.onDataChanged

### LookupDesign

**Hierarchy**

- InputControlDesign
  - LookupDesign

### Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>alignItems</td>
<td>alignItems: string (optional)</td>
<td>This property is an alias for the CSS property &quot;align-items&quot;. Inherited from Design.alignItems</td>
</tr>
<tr>
<td>alignSelf</td>
<td>alignSelf: string (optional)</td>
<td>Inherited from Design.alignSelf</td>
</tr>
<tr>
<td>bindings</td>
<td>bindings: any (optional)</td>
<td>Inherited from Design.bindings</td>
</tr>
<tr>
<td>border</td>
<td>border: &quot;none&quot;</td>
<td>&quot;solid&quot;</td>
</tr>
<tr>
<td>color</td>
<td>color: string (optional)</td>
<td>The foreground color of the container. Inherited from Design.color</td>
</tr>
</tbody>
</table>
### flexFlow

**Signature**: `flexFlow: string (optional)`  
**Description**: Specifying this property makes the component a flex container component. Inherited from `Design.flexFlow`

### flexSize

**Signature**: `flexSize: string (optional)`  
**Description**: One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. Inherited from `Design.flexSize`

### fontSize

**Signature**: `fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)`  
**Description**: The proportional text size. Inherited from `Design.fontSize`

### fontWeight

**Signature**: `fontWeight: "normal" | "bold" (optional)`  
**Description**: Normal or bold text. Inherited from `Design.fontWeight`

### justifyItems

**Signature**: `justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)`  
**Description**: This property is an alias for the CSS property "justify-content". Inherited from `Design.justifyItems`

### label

**Signature**: `label: string (optional)`  
**Description**: Inherited from `Design.label`

### labelPosition

**Signature**: `labelPosition: "stacked" | "hidden" | "inline" (optional)`  
**Description**: Determines how a label is positioned, if at all. By default, labelPosition is set to stacked. Inherited from `Design.labelPosition`

### name

**Signature**: `name: string (optional)`  
**Description**: Inherited from `Design.name`

### padding

**Signature**: `padding: "none" | "small" | "std" (optional)`  
**Description**: Allows specifying the component's padding behavior. Inherited from `Design.padding`

### type

**Signature**: `type: ControlType (optional)`  
**Description**: The type of the control as a string. Inherited from `Design.type`

---

**LookupMetadata**

**Hierarchy**

- **InputControlMetadata**
  - `LookupMetadata`

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoundEntity</td>
<td><code>BoundEntity: string (optional)</code></td>
<td>The entity to which the control is bound. Inherited from <code>ControlMetadata.BoundEntity</code></td>
</tr>
<tr>
<td>BoundField</td>
<td><code>BoundField: string (optional)</code></td>
<td>Inherited from <code>ControlMetadata.BoundField</code></td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Description</td>
<td>Description: string (optional)</td>
<td>Description of the control. Inherited from ControlMetadata.Description</td>
</tr>
<tr>
<td>DisplayField</td>
<td>DisplayField: string (optional)</td>
<td>The name of a control on the page, whose value should be displayed to the user. Usually, this value is user-friendly user-readable text.</td>
</tr>
<tr>
<td>DisplayKey</td>
<td>DisplayKey: string (optional)</td>
<td></td>
</tr>
<tr>
<td>Editable</td>
<td>Editable: boolean (optional)</td>
<td>Boolean indicating if the control is editable. Inherited from ControlMetadataEditable</td>
</tr>
<tr>
<td>ExtType</td>
<td>ExtType: ControlType (optional)</td>
<td>The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType</td>
</tr>
<tr>
<td>FilterContext</td>
<td>FilterContext: DataFilter (optional)</td>
<td></td>
</tr>
<tr>
<td>HelpText</td>
<td>HelpText: string (optional)</td>
<td>The keyboard shortcut for a command. For example, &quot;(Shift+F5)&quot; Inherited from ControlMetadata.HelpText</td>
</tr>
<tr>
<td>Hidden</td>
<td>Hidden: boolean (optional)</td>
<td>Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden</td>
</tr>
<tr>
<td>Id</td>
<td>Id: string (optional)</td>
<td>Identification string for a control. Inherited from ControlMetadata.Id</td>
</tr>
<tr>
<td>Label</td>
<td>Label: string (optional)</td>
<td>Label for a control. For example, a control representing a person's first name might have a label &quot;First Name&quot;. Inherited from ControlMetadata.Label</td>
</tr>
<tr>
<td>LookupEntity</td>
<td>LookupEntity: any (optional)</td>
<td>The entity that is being looked up in the lookup.</td>
</tr>
<tr>
<td>LookupPage</td>
<td>LookupPage: string (optional)</td>
<td></td>
</tr>
<tr>
<td>LookupPageld</td>
<td>LookupPageld: string (optional)</td>
<td></td>
</tr>
<tr>
<td>Mandatory</td>
<td>Mandatory: boolean (optional)</td>
<td>If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline. Inherited from InputControlMetadata.Mandatory</td>
</tr>
<tr>
<td>Name</td>
<td>Signature</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MultiSelect</td>
<td>MultiSelect: boolean (optional)</td>
<td>If true, lookup will be configured as a multi-select.</td>
</tr>
<tr>
<td>Name</td>
<td>Name: string (optional)</td>
<td>Name of a control. Inherited from ControlMetadata.Name</td>
</tr>
<tr>
<td>NumSequence</td>
<td>NumSequence: NumberSequenceConfig (optional)</td>
<td>Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Inherited from InputControlMetadata.NumSequence</td>
</tr>
<tr>
<td>Order</td>
<td>Order: number (optional)</td>
<td>Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order</td>
</tr>
<tr>
<td>ReferenceAppId</td>
<td>ReferenceAppId: string (optional)</td>
<td></td>
</tr>
<tr>
<td>ShowLookupPage</td>
<td>ShowLookupPage: boolean (optional)</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Type: ControlType (optional)</td>
<td>String indicating the control type. Inherited from ControlMetadata.Type</td>
</tr>
<tr>
<td>ValueField</td>
<td>ValueField: string (optional)</td>
<td>The name of a control on the page, whose value should be used when committing the data. Usually, this value is a unique key.</td>
</tr>
<tr>
<td>ValueKey</td>
<td>ValueKey: string (optional)</td>
<td></td>
</tr>
</tbody>
</table>

| Events            |                                               |                                                                           |
|-------------------|-----------------------------------------------|                                                                          |
| Name              | Signature                                      | Description                                                                 |
| OnOptionSelected  | OnOptionSelected: function(lookup: any, lookupEntityData: any): void (optional) | An event that is triggered by an option being selected.                   |
| OnValueChanged    | OnValueChanged: function(value: any): void (optional) | An event that is triggered by a value being changed.                     |
Multi-Lookup controls are similar to regular lookups except they allow multiple selections at once.

Index

Types
- MultiLookup
- MultiLookupDesign
- MultiLookupMetadata

Types
MultiLookup

Hierarchy
InputControl
  └ MultiLookup

Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>container: boolean (optional)</td>
<td>True if the control is a container. Inherited from Control.container</td>
</tr>
<tr>
<td>generic</td>
<td>generic: boolean (optional)</td>
<td>Inherited from Control.generic</td>
</tr>
<tr>
<td>getDataSource</td>
<td>getDataSource: function(): any</td>
<td>Inherited from Control.getDataSource</td>
</tr>
<tr>
<td>getEntityRefs</td>
<td>getEntityRefs: function(): string []</td>
<td>Inherited from MultiLookupMetadata</td>
</tr>
<tr>
<td>hidden</td>
<td>hidden: boolean</td>
<td>True if the control is hidden. Inherited from Control.hidden</td>
</tr>
<tr>
<td>setEntityRefs</td>
<td>setEntityRefs: function(ids: string []</td>
<td>Promise &lt;any&gt;</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>applyDesign</td>
<td>applyDesign(IDesign: MultiLookupDesign): void</td>
<td>Applies given design to the design on the control. Overrides Control.applyDesign</td>
</tr>
<tr>
<td>dataSource</td>
<td>dataSource(): any</td>
<td>Inherited from Control.dataSource</td>
</tr>
</tbody>
</table>

Multi-Lookup controls are similar to regular lookups except they allow multiple selections at once.
### getDesign

**getDesign()**: `Design`  
Returns the design object of this control. Inherited from `Control.getDesign`.

### getLookupPage

**getLookupPage()**: `Page`  

### isEditable

**isEditable()**: `boolean`  
Boolean indicating if the control is editable. Inherited from `Control.isEditable`.

### metadata

**metadata()**: `MultiLookupMetadata`  
Returns the metadata object of this control. Overrides `InputControl.metadata`.

### parent

**parent()**: `Control | Page`  
Returns the parent (control or page) of this control. Inherited from `Control.parent`.

### root

**root()**: `Page`  
Returns the root form instance (page) of this control. Inherited from `Control.root`.

### Events

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>onDataChange</td>
<td>onDataChange: EventHook</td>
<td>An event that is triggered when the input control's data changes. Inherited from <code>InputControl.onDataChanged</code></td>
</tr>
</tbody>
</table>

### MultiLookupDesign

**Hierarchy**  
`InputControlDesign`  
  └ MultiLookupDesign

### Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>alignItems</td>
<td>alignItems: string (optional)</td>
<td>This property is an alias for the CSS property &quot;align-items&quot;. Inherited from <code>Design.alignItems</code></td>
</tr>
<tr>
<td>alignSelf</td>
<td>alignSelf: string (optional)</td>
<td>Inherited from <code>Design.alignSelf</code></td>
</tr>
<tr>
<td>bindings</td>
<td>bindings: any (optional)</td>
<td>Inherited from <code>Design.bindings</code></td>
</tr>
<tr>
<td>border</td>
<td>border: &quot;none&quot;</td>
<td>&quot;solid&quot;</td>
</tr>
<tr>
<td>color</td>
<td>color: string (optional)</td>
<td>The foreground color of the container. Inherited from <code>Design.color</code></td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>flexFlow</td>
<td>flexFlow: string (optional)</td>
<td>Specifying this property makes the component a flex container component. Inherited from Design.flexFlow</td>
</tr>
<tr>
<td>flexSize</td>
<td>flexSize: string (optional)</td>
<td>One number or two numbers written as a string. For example, &quot;(size to grow) [(size-to-shrink)]&quot; to accommodate available space in the immediate flex container. Inherited from Design.flexSize</td>
</tr>
<tr>
<td>fontSize</td>
<td>fontSize: &quot;medium&quot;</td>
<td>&quot;xx-small&quot;</td>
</tr>
<tr>
<td>fontWeight</td>
<td>fontWeight: &quot;normal&quot;</td>
<td>&quot;bold&quot; (optional)</td>
</tr>
<tr>
<td>justifyItems</td>
<td>justifyItems: &quot;flex-start&quot;</td>
<td>&quot;flex-end&quot;</td>
</tr>
<tr>
<td>label</td>
<td>label: string (optional)</td>
<td>Inherited from Design.label</td>
</tr>
<tr>
<td>labelPosition</td>
<td>labelPosition: &quot;stacked&quot;</td>
<td>&quot;hidden&quot;</td>
</tr>
<tr>
<td>name</td>
<td>name: string (optional)</td>
<td>Inherited from Design.name</td>
</tr>
<tr>
<td>padding</td>
<td>padding: &quot;none&quot;</td>
<td>&quot;small&quot;</td>
</tr>
<tr>
<td>type</td>
<td>type: ControlType (optional)</td>
<td>The type of the control as a string. Inherited from Design.type</td>
</tr>
</tbody>
</table>

**MultiLookupMetadata**

**Hierarchy**

- InputControlMetadata
  - MultiLookupMetadata

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoundEntity</td>
<td>BoundEntity: string (optional)</td>
<td>The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity</td>
</tr>
<tr>
<td>BoundField</td>
<td>BoundField: string (optional)</td>
<td>Inherited from ControlMetadata.BoundField</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Description</td>
<td>Description: string (optional)</td>
<td>Description of the control. Inherited from ControlMetadata.Description</td>
</tr>
<tr>
<td>Design</td>
<td>Design: Design (optional)</td>
<td>Design object for the lookup page that is referenced by the LookupPageld.</td>
</tr>
<tr>
<td>Editable</td>
<td>Editable: boolean (optional)</td>
<td>Boolean indicating if the control is editable. Inherited from ControlMetadata.Editable</td>
</tr>
<tr>
<td>ExtType</td>
<td>ExtType: ControlType (optional)</td>
<td>The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType</td>
</tr>
<tr>
<td>FilterContext</td>
<td>FilterContext: DataFilter (optional)</td>
<td></td>
</tr>
<tr>
<td>FilterLocalOnly</td>
<td>FilterLocalOnly: boolean (optional)</td>
<td></td>
</tr>
<tr>
<td>HelpText</td>
<td>HelpText: string (optional)</td>
<td>The keyboard shortcut for a command. For example, &quot;;(Shift+F5)&quot; Inherited from ControlMetadata.HelpText</td>
</tr>
<tr>
<td>Hidden</td>
<td>Hidden: boolean (optional)</td>
<td>Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden</td>
</tr>
<tr>
<td>Id</td>
<td>Id: string (optional)</td>
<td>Identification string for a control. Inherited from ControlMetadata.Id</td>
</tr>
<tr>
<td>Label</td>
<td>Label: string (optional)</td>
<td>Label for a control. For example, a control representing a person's first name might have a label &quot;First Name&quot;. Inherited from ControlMetadata.Label</td>
</tr>
<tr>
<td>LookupPageld</td>
<td>LookupPageld: string (optional)</td>
<td>Page that is hosted within the multi-lookup.</td>
</tr>
<tr>
<td>Mandatory</td>
<td>Mandatory: boolean (optional)</td>
<td>If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline. Inherited from InputControlMetadata.Mandatory</td>
</tr>
<tr>
<td>Name</td>
<td>Name: string (optional)</td>
<td>Name of a control. Inherited from ControlMetadata.Name</td>
</tr>
</tbody>
</table>
### NumSequence

**NAME**: NumSequence

**SIGNATURE**: NumSequence: NumberSequenceConfig (optional)

**DESCRIPTION**: Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Inherited from InputControlMetadata.NumSequence

### Order

**NAME**: Order

**SIGNATURE**: Order: number (optional)

**DESCRIPTION**: Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order

### ReferenceAppId

**NAME**: ReferenceAppId

**SIGNATURE**: ReferenceAppId: string (optional)

### ReverseLookupRelation

**NAME**: ReverseLookupRelation

**SIGNATURE**: ReverseLookupRelation: boolean (optional)

### ShowPending

**NAME**: ShowPending

**SIGNATURE**: ShowPending: boolean (optional)

### Type

**NAME**: Type

**SIGNATURE**: Type: ControlType (optional)

**DESCRIPTION**: String indicating the control type. Inherited from ControlMetadata.Type

### Events

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnLookupPageCreated</td>
<td>OnLookupPageCreated: function(args: any, multiLookup: any): void (optional)</td>
<td></td>
</tr>
</tbody>
</table>
The IPage interface encapsulates the various properties, life cycle and event hooks associated with a page in a workspace.

**Page data Synchronization**

Pages that can submit changes (also referred to as actions) go through various stages before they’re completely in sync with the server. As soon as a page is submitted, there are three possible consequences:

- It fails client-side validation: The client logic might prevent the page from being submitted.
- The client is online: The submission is processed as soon as the application-wide sync queue is cleared.
- The client is offline: The submission is added to the application-wide sync queue and stays there as long as the client is offline.

While a submission is waiting to be synchronized, it can be in one of these states:

- Pending: The submission is still pending and can still be edited.
- Processing: The submission is currently being synchronized. In this state, any further edits are not allowed on the page.

After a submission goes through to the server, it can be in one of these states:

- Synchronized: The submission is accepted by the server and is synchronized.
- Error: The server rejects the submission and the page enters an error state.

Multiple pending submissions for a given page and context might be clubbed (and sent) together if the page is deemed to be idempotent. This implies that the server need not process all submissions in any order and depends on the design of the page on the server.

**Index**

**Enumerations**
- PageState

**Types**
- CompleteEventArgs
- Design
- NavigationArgs
- Page
- PageMetadata
- PageOptions
- PageSubmitArgs
- PageTarget

**Enumerations**

**PageState**

Enumeration members
<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>error</td>
<td>error: boolean (optional)</td>
<td>The page is currently in the error state, but can be refreshed in an attempt to get out of this state.</td>
</tr>
<tr>
<td>loaded</td>
<td>loaded: boolean (optional)</td>
<td>The page is fully loaded and can be refreshed and, if possible, submitted.</td>
</tr>
<tr>
<td>loading</td>
<td>loading: boolean (optional)</td>
<td>The page is currently being loaded.</td>
</tr>
<tr>
<td>offline</td>
<td>offline: boolean (optional)</td>
<td>The page was loaded in the offline mode, thus not refreshable.</td>
</tr>
<tr>
<td>refreshing</td>
<td>refreshing: boolean (optional)</td>
<td>The page is currently refreshing its data.</td>
</tr>
</tbody>
</table>

**Types**

**CompleteEventArgs**

**Hierarchy**

CompleteEventArgs

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>error</td>
<td>error: boolean (optional)</td>
<td></td>
</tr>
<tr>
<td>navigation</td>
<td>navigation: NavigationArgs (optional)</td>
<td></td>
</tr>
<tr>
<td>processed</td>
<td>processed: boolean (optional)</td>
<td></td>
</tr>
</tbody>
</table>

**Design**

**Hierarchy**

Design

- PageLinkDesign
- ContainerControlDesign
- InputControlDesign
- ImageDesign

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>alignItems</td>
<td>alignItems: string (optional)</td>
<td>This property is an alias for the CSS property “align-items”.</td>
</tr>
<tr>
<td>alignSelf</td>
<td>alignSelf: string (optional)</td>
<td></td>
</tr>
<tr>
<td>bindings</td>
<td>bindings: any (optional)</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>border</td>
<td>border: &quot;none&quot;</td>
<td>&quot;solid&quot;</td>
</tr>
<tr>
<td>color</td>
<td>color: string (optional)</td>
<td>The foreground color of the container.</td>
</tr>
<tr>
<td>flexFlow</td>
<td>flexFlow: string (optional)</td>
<td>Specifying this property makes the component a flex container component.</td>
</tr>
<tr>
<td>flexSize</td>
<td>flexSize: string (optional)</td>
<td>One number or two numbers written as a string. For example, &quot;(size to grow) ([size-to-shrink])&quot; to accommodate available space in the immediate flex container.</td>
</tr>
<tr>
<td>fontSize</td>
<td>fontSize: &quot;medium&quot;</td>
<td>&quot;xx-small&quot;</td>
</tr>
<tr>
<td>fontWeight</td>
<td>fontWeight: &quot;normal&quot;</td>
<td>&quot;bold&quot; (optional)</td>
</tr>
<tr>
<td>justifyItems</td>
<td>justifyItems: &quot;flex-start&quot;</td>
<td>&quot;flex-end&quot;</td>
</tr>
<tr>
<td>label</td>
<td>label: string (optional)</td>
<td></td>
</tr>
<tr>
<td>labelPosition</td>
<td>labelPosition: &quot;stacked&quot;</td>
<td>&quot;hidden&quot;</td>
</tr>
<tr>
<td>name</td>
<td>name: string (optional)</td>
<td></td>
</tr>
<tr>
<td>padding</td>
<td>padding: &quot;none&quot;</td>
<td>&quot;small&quot;</td>
</tr>
<tr>
<td>type</td>
<td>type: ControlType (optional)</td>
<td>The type of the control as a string.</td>
</tr>
</tbody>
</table>

**NavigationArgs**

**Hierarchy**

PageTarget
  └ NavigationArgs

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>label</td>
<td>label: string (optional)</td>
<td></td>
</tr>
<tr>
<td>options</td>
<td>options: any (optional)</td>
<td></td>
</tr>
<tr>
<td>params</td>
<td>params: PageOptions (optional)</td>
<td>Inherited from PageTarget, params</td>
</tr>
</tbody>
</table>
### Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>replace</td>
<td>replace: boolean (optional)</td>
<td>If set to true, removes current view firing navigation from navigation history stack.</td>
</tr>
<tr>
<td>to</td>
<td>to: string (optional)</td>
<td>Inherited from PageTarget.to</td>
</tr>
<tr>
<td>url</td>
<td>url: string (optional)</td>
<td>If provided, this link is directly opened.</td>
</tr>
</tbody>
</table>

#### Page

#### Hierarchy

#### Page

#### Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>children</td>
<td>children: Control [ ]</td>
<td>The list of all direct children controls of the page.</td>
</tr>
<tr>
<td>dataLoadedInitially</td>
<td>dataLoadedInitially: Promise &lt;void&gt;</td>
<td>A promise which resolves when the data has loaded for the first time.</td>
</tr>
<tr>
<td>initialized</td>
<td>initialized: boolean</td>
<td>True if the page instance has been initialized.</td>
</tr>
<tr>
<td>metadata</td>
<td>metadata: PageMetadata</td>
<td>The page metadata.</td>
</tr>
<tr>
<td>metadataLoaded</td>
<td>metadataLoaded: Promise &lt;void&gt;</td>
<td>A promise which resolves when the metadata has finished loading.</td>
</tr>
<tr>
<td>pageContext</td>
<td>pageContext: string</td>
<td>The current page context.</td>
</tr>
<tr>
<td>pageFilter</td>
<td>pageFilter: DataFilter</td>
<td>The current filter applied on the page.</td>
</tr>
<tr>
<td>state</td>
<td>state: PageState</td>
<td>The current state of the page.</td>
</tr>
<tr>
<td>syncError</td>
<td>syncError: boolean</td>
<td>True if the page's submission is in error state. This normally happens when the server rejects submissions due to validation errors.</td>
</tr>
<tr>
<td>syncPending</td>
<td>syncPending: boolean</td>
<td>True if the page's submission is waiting to be synced.</td>
</tr>
<tr>
<td>syncProcessing</td>
<td>syncProcessing: boolean</td>
<td>True if the page instance is currently syncing its submission.</td>
</tr>
<tr>
<td>syncUnitEditable</td>
<td>syncUnitEditable: boolean</td>
<td>True if it's possible to edit a submission while it's waiting to be synchronized.</td>
</tr>
<tr>
<td>title</td>
<td>title: string</td>
<td>The title of the page.</td>
</tr>
</tbody>
</table>

#### Methods
<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>canSubmit</td>
<td>canSubmit(): boolean</td>
<td>Returns true if action page can be submitted and there are no validation error messages.</td>
</tr>
<tr>
<td>close</td>
<td>close(): void</td>
<td>Dispose the page instance and all its lifecycle events.</td>
</tr>
<tr>
<td>getAction</td>
<td>getAction(actionName: string): PageLink</td>
<td>Get a page action by name.</td>
</tr>
<tr>
<td>getActions</td>
<td>getActions(): PageLink []</td>
<td>Get all page actions.</td>
</tr>
<tr>
<td>getControl</td>
<td>getControl(controlName: string): Control</td>
<td>Get a page control by name.</td>
</tr>
<tr>
<td>getDesign</td>
<td>getDesign(): Design</td>
<td>Get the design object associated with the page.</td>
</tr>
<tr>
<td>getEntityContext</td>
<td>getEntityContext(): EntityRef</td>
<td>Get current entity context.</td>
</tr>
<tr>
<td>isEditable</td>
<td>isEditable(): boolean</td>
<td>Returns true if the page is an Action Page</td>
</tr>
<tr>
<td>refreshData</td>
<td>refreshData(): Promise &lt;void&gt;</td>
<td>Force refresh page data.</td>
</tr>
<tr>
<td>resume</td>
<td>resume(): Promise &lt;void&gt;</td>
<td>Resume a temporarily suspended page.</td>
</tr>
<tr>
<td>submit</td>
<td>submit(): Promise &lt;CompleteEventArgs&gt;</td>
<td>Submit an Action.</td>
</tr>
<tr>
<td>suspend</td>
<td>suspend(): void</td>
<td>Temporarily suspend a page.</td>
</tr>
</tbody>
</table>

**Events**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>onClose</td>
<td>onClose: EventHook &lt;null&gt;</td>
<td>Event that is raised when a page is closed.</td>
</tr>
<tr>
<td>onComplete</td>
<td>onComplete: EventHook &lt;any&gt;</td>
<td>Event that is raised when an action is completed.</td>
</tr>
<tr>
<td>onDataLoaded</td>
<td>onDataLoaded: EventHook &lt;any&gt;</td>
<td>Event that fires when the page data has loaded.</td>
</tr>
<tr>
<td>onInit</td>
<td>onInit: EventHook &lt;any&gt;</td>
<td>Event that fires when a page instance has been initialized, and the metadata has been loaded.</td>
</tr>
<tr>
<td>onPreInit</td>
<td>onPreInit: EventHook &lt;any&gt;</td>
<td>Event that fires when a page instance has been initialized.</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>onRefresh</td>
<td>onRefresh: EventHook &lt;null&gt;</td>
<td>Event that fires on forced page refresh, before new data has been loaded.</td>
</tr>
<tr>
<td>onStateChange</td>
<td>onStateChange: EventHook &lt;null&gt;</td>
<td>Event that fires when the page state changes.</td>
</tr>
<tr>
<td>onSubmit</td>
<td>onSubmit: EventHook &lt;PageSubmitArgs&gt;</td>
<td>Event that fires before an action is submitted. It can be intercepted for</td>
</tr>
<tr>
<td></td>
<td></td>
<td>action validation deferring.</td>
</tr>
<tr>
<td>onSyncStatusChange</td>
<td>onSyncStatusChange: EventHook &lt;null&gt;</td>
<td>Event that fires when the page sync status changes.</td>
</tr>
</tbody>
</table>

**PageMetadata**

**Hierarchy**

**PageMetadata**

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controls</td>
<td>Controls: ControlMetadata [] (optional)</td>
<td></td>
</tr>
<tr>
<td>Design</td>
<td>Design: Design (optional)</td>
<td></td>
</tr>
<tr>
<td>Id</td>
<td>Id: string (optional)</td>
<td></td>
</tr>
<tr>
<td>QuickSubmit</td>
<td>QuickSubmit: boolean (optional)</td>
<td></td>
</tr>
<tr>
<td>SourcePageId</td>
<td>SourcePageId: string (optional)</td>
<td></td>
</tr>
<tr>
<td>SubmitButtonDesign</td>
<td>SubmitButtonDesign: Design (optional)</td>
<td></td>
</tr>
<tr>
<td>Tasks</td>
<td>Tasks: PageMetadata [] (optional)</td>
<td></td>
</tr>
<tr>
<td>Title</td>
<td>Title: string (optional)</td>
<td></td>
</tr>
</tbody>
</table>

**Events**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnDataLoaded</td>
<td>OnDataLoaded: function(sender: Page, dataWrapper: any): void (optional)</td>
<td></td>
</tr>
<tr>
<td>OnInit</td>
<td>OnInit: function(sender: Page): void (optional)</td>
<td></td>
</tr>
<tr>
<td>OnPreInit</td>
<td>OnPreInit: function(sender: Page): void (optional)</td>
<td></td>
</tr>
<tr>
<td>OnSubmit</td>
<td>OnSubmit: function(dataValues: any, args: any): void (optional)</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>OnTaskSubmitted</td>
<td>OnTaskSubmitted: function(taskHandle: any, taskOptions: any): any (optional)</td>
<td></td>
</tr>
<tr>
<td>OnTaskSubmitting</td>
<td>OnTaskSubmitting: function(taskOptions: any): any (optional)</td>
<td></td>
</tr>
</tbody>
</table>

**PageOptions**

**Hierarchy**

**PageOptions**

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>appId</td>
<td>appId: string (optional)</td>
<td></td>
</tr>
<tr>
<td>design</td>
<td>design: Design (optional)</td>
<td></td>
</tr>
<tr>
<td>excludeContext</td>
<td>excludeContext: boolean (optional)</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>filter: DataFilter (optional)</td>
<td></td>
</tr>
<tr>
<td>filterLocalOnly</td>
<td>filterLocalOnly: boolean (optional)</td>
<td></td>
</tr>
<tr>
<td>pageContext</td>
<td>pageContext: string (optional)</td>
<td></td>
</tr>
<tr>
<td>pageId</td>
<td>pageId: string (optional)</td>
<td></td>
</tr>
<tr>
<td>readOptions</td>
<td>readOptions: IReadOptions (optional)</td>
<td></td>
</tr>
</tbody>
</table>

**PageSubmitArgs**

**Hierarchy**

**PageSubmitArgs**

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataValues</td>
<td>dataValues: any</td>
<td>Get the payload of the submit action.</td>
</tr>
<tr>
<td>sender</td>
<td>sender: Page</td>
<td>Get the sender page instance of the submit action.</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>addMessage</td>
<td>addMessage(message: string, type: any): any</td>
<td>Add a validation error message to be displayed.</td>
</tr>
<tr>
<td>cancel</td>
<td>cancel(): any</td>
<td>Prevent the action from submitting.</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>getMessages</td>
<td>getMessages(): string []</td>
<td>Get all previously added messages</td>
</tr>
<tr>
<td>isCancelled</td>
<td>isCancelled(): boolean</td>
<td>Check if the submit action is cancelled.</td>
</tr>
<tr>
<td>wait</td>
<td>wait(promise: Promise &lt;any&gt;): any</td>
<td>Wait on a given promise before continuing with the submission.</td>
</tr>
</tbody>
</table>

**PageTarget**

**Hierarchy**

PageTarget

└— NavigationArgs

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>params</td>
<td>params: PageOptions (optional)</td>
<td></td>
</tr>
<tr>
<td>to</td>
<td>to: string (optional)</td>
<td></td>
</tr>
</tbody>
</table>
A pagelink is a control that navigates to another page.

Index

Types
- PageLink
- PageLinkDesign
- PageLinkMetadata

Types

PageLink

Hierarchy
Control
  └── PageLink

Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>container: boolean (optional)</td>
<td>True if the control is a container. Inherited from Control.container</td>
</tr>
<tr>
<td>generic</td>
<td>generic: boolean (optional)</td>
<td>Inherited from Control.generic</td>
</tr>
<tr>
<td>getDataSource</td>
<td>getDataSource: function(): any</td>
<td>Inherited from Control.getDataSource</td>
</tr>
<tr>
<td>hidden</td>
<td>hidden: boolean</td>
<td>True if the control is hidden. Inherited from Control.hidden</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>allowsNavigation</td>
<td>allowsNavigation(): boolean</td>
<td></td>
</tr>
<tr>
<td>applyDesign</td>
<td>applyDesign(design: PageLinkDesign):</td>
<td>Applies given design to the design on the control. Overrides Control.applyDesign</td>
</tr>
<tr>
<td></td>
<td>void</td>
<td></td>
</tr>
<tr>
<td>dataContext</td>
<td>dataContext(): any</td>
<td>Inherited from Control.dataContext</td>
</tr>
<tr>
<td>getCount</td>
<td>getCount(): number</td>
<td>string</td>
</tr>
<tr>
<td>getDesign</td>
<td>getDesign(): Design</td>
<td>Returns the design object of this control. Inherited from Control.getDesign</td>
</tr>
</tbody>
</table>

A pagelink is a control that navigates to another page.
<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>getNavigationHandler</td>
<td>getNavigationHandler(): NavigationArgs</td>
<td></td>
</tr>
<tr>
<td>isEditable</td>
<td>isEditable(): boolean</td>
<td>Boolean indicating if the control is editable. Inherited from Control.isEditable</td>
</tr>
<tr>
<td>metadata</td>
<td>metadata(): PageLinkMetadata</td>
<td>Returns the metadata object of this control. Overrides Control.metadata</td>
</tr>
<tr>
<td>parent</td>
<td>parent(): Control</td>
<td>Page</td>
</tr>
<tr>
<td>root</td>
<td>root(): Page</td>
<td>Returns the root form instance (page) of this control. Inherited from Control.root</td>
</tr>
<tr>
<td>showCount</td>
<td>showCount(): boolean</td>
<td></td>
</tr>
</tbody>
</table>

**PageLinkDesign**

**Hierarchy**

Design
    ├─ PageLinkDesign

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>alignItems</td>
<td>alignItems: string (optional)</td>
<td>This property is an alias for the CSS property &quot;align-items&quot;. Inherited from Design.alignItems</td>
</tr>
<tr>
<td>alignSelf</td>
<td>alignSelf: string (optional)</td>
<td>Inherited from Design.alignSelf</td>
</tr>
<tr>
<td>background</td>
<td>background: string (optional)</td>
<td>Sets the background color</td>
</tr>
<tr>
<td>bindings</td>
<td>bindings: any (optional)</td>
<td>Inherited from Design.bindings</td>
</tr>
<tr>
<td>border</td>
<td>border: &quot;none&quot;</td>
<td>&quot;solid&quot;</td>
</tr>
<tr>
<td>color</td>
<td>color: string (optional)</td>
<td>The foreground color of the container. Inherited from Design.color</td>
</tr>
<tr>
<td>excludeContext</td>
<td>excludeContext: boolean (optional)</td>
<td></td>
</tr>
<tr>
<td>flexFlow</td>
<td>flexFlow: string (optional)</td>
<td>Specifying this property makes the component a flex container component. Inherited from Design.flexFlow</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>flexSize</td>
<td>flexSize: string (optional)</td>
<td>One number or two numbers written as a string. For example, &quot;(size to grow) [(size-to-shrink)]&quot; to accommodate available space in the immediate flex container. Inherited from Design.flexSize</td>
</tr>
<tr>
<td>fontSize</td>
<td>fontSize: &quot;medium&quot;</td>
<td>The proportional text size Inherited from Design.fontSize</td>
</tr>
<tr>
<td>fontWeight</td>
<td>fontWeight: &quot;normal&quot;</td>
<td>Normal or bold text. Inherited from Design.fontSize</td>
</tr>
<tr>
<td>hideArrow</td>
<td>hideArrow: boolean (optional)</td>
<td>Allows an arrow ( &gt; ) on a default styled navigation control to be hidden.</td>
</tr>
<tr>
<td>icon</td>
<td>icon: string (optional)</td>
<td>Name of the icon that is displayed in the pagelink control.</td>
</tr>
<tr>
<td>justifyItems</td>
<td>justifyItems: &quot;flex-start&quot;</td>
<td>This property is an alias for the CSS property &quot;justify-content&quot;. Inherited from Design.justifyItems</td>
</tr>
<tr>
<td>label</td>
<td>label: string (optional)</td>
<td>Inherited from Design.label</td>
</tr>
<tr>
<td>labelPosition</td>
<td>labelPosition: &quot;stacked&quot;</td>
<td>Determines how a label is positioned, if at all. By default, labelPosition is set to stacked. Inherited from Design.labelPosition</td>
</tr>
<tr>
<td>name</td>
<td>name: string (optional)</td>
<td>Inherited from Design.name</td>
</tr>
<tr>
<td>padding</td>
<td>padding: &quot;none&quot;</td>
<td>Allows specifying the component's padding behavior. Inherited from Design.padding</td>
</tr>
<tr>
<td>showCount</td>
<td>showCount: boolean (optional)</td>
<td>If true, shows a count of the records present in the list on the target page.</td>
</tr>
<tr>
<td>style</td>
<td>style: string (optional)</td>
<td>Determines the visual style of the pagelink control. Options: * &quot;inline&quot;: takes up the full width its container, with the label in-line with the icon * &quot;button&quot;: takes up only as much width as needed by the label, with the label below the icon</td>
</tr>
<tr>
<td>type</td>
<td>type: ControlType (optional)</td>
<td>The type of the control as a string. Inherited from Design.type</td>
</tr>
</tbody>
</table>
### Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoundEntity</td>
<td><code>BoundEntity: string (optional)</code></td>
<td>The entity to which the control is bound. Inherited from <code>ControlMetadata.BoundEntity</code></td>
</tr>
<tr>
<td>BoundField</td>
<td><code>BoundField: string (optional)</code></td>
<td>Inherited from <code>ControlMetadata.BoundField</code></td>
</tr>
<tr>
<td>Description</td>
<td><code>Description: string (optional)</code></td>
<td>Description of the control. Inherited from <code>ControlMetadata.Description</code></td>
</tr>
<tr>
<td>Editable</td>
<td><code>Editable: boolean (optional)</code></td>
<td>Boolean indicating if the control is editable. Inherited from <code>ControlMetadata.Ediable</code></td>
</tr>
<tr>
<td>ExcludeContext</td>
<td><code>ExcludeContext: boolean (optional)</code></td>
<td></td>
</tr>
<tr>
<td>ExtType</td>
<td><code>ExtType: ControlType (optional)</code></td>
<td>The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from <code>ControlMetadata.ExtType</code></td>
</tr>
<tr>
<td>HelpText</td>
<td><code>HelpText: string (optional)</code></td>
<td>The keyboard shortcut for a command. For example, &quot;(Shift+F5)&quot; Inherited from <code>ControlMetadata.HelpText</code></td>
</tr>
<tr>
<td>Hidden</td>
<td><code>Hidden: boolean (optional)</code></td>
<td>Boolean indicating if the control is hidden or not. Inherited from <code>ControlMetadata.Hidden</code></td>
</tr>
<tr>
<td>Icon</td>
<td><code>Icon: string (optional)</code></td>
<td>Name of the icon that is displayed in the pagelink control.</td>
</tr>
<tr>
<td>IconSize</td>
<td><code>IconSize: number (optional)</code></td>
<td>Determines the size of the icon that is displayed in the pagelink control.</td>
</tr>
<tr>
<td>Id</td>
<td><code>Id: string (optional)</code></td>
<td>Identification string for a control. Inherited from <code>ControlMetadata.Id</code></td>
</tr>
<tr>
<td>Label</td>
<td><code>Label: string (optional)</code></td>
<td>Label for a control. For example, a control representing a person’s first name might have a label “First Name”. Inherited from <code>ControlMetadata.Label</code></td>
</tr>
<tr>
<td>Name</td>
<td><code>Name: string (optional)</code></td>
<td>Name of a control. Inherited from <code>ControlMetadata.Name</code></td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Order</td>
<td>Order: number (optional)</td>
<td>Number indicating the order in which a control will appear on a page.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inherited from ControlMetadata.Order</td>
</tr>
<tr>
<td>ShowCount</td>
<td>ShowCount: boolean (optional)</td>
<td>If true, shows a count of the records present in the list on the target page.</td>
</tr>
<tr>
<td>Style</td>
<td>Style: string (optional)</td>
<td>Determines the visual style of the pagelink control.</td>
</tr>
<tr>
<td>Target</td>
<td>Target: string (optional)</td>
<td>Name of the target action or page to navigate to when the pagelink is selected.</td>
</tr>
<tr>
<td>Type</td>
<td>Type: ControlType (optional)</td>
<td>String indicating the control type. Inherited from ControlMetadata.Type</td>
</tr>
<tr>
<td>UseDataContext</td>
<td>UseDataContext: boolean (optional)</td>
<td></td>
</tr>
</tbody>
</table>

**Events**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnNavigate</td>
<td>OnNavigate: function(navigation: NavigationArgs</td>
<td>An event that is triggered when the navigation is triggered.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>string: any (optional))</td>
</tr>
</tbody>
</table>
A part is a container control that contains only a page, allowing for a page to be embedded within a page.

Index

Types

- Part
- PartDesign
- PartMetadata

Types

Part

Hierarchy

ContainerControl

└─ Part

Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>container: boolean</td>
<td>True if the control is a container. Inherited from ContainerControl.container. Overrides Control.container</td>
</tr>
<tr>
<td>generic</td>
<td>generic: boolean (optional)</td>
<td>Inherited from Control.generic</td>
</tr>
<tr>
<td>getDataSource</td>
<td>getDataSource: function(): any</td>
<td>Inherited from Control.getDataSource</td>
</tr>
<tr>
<td>hidden</td>
<td>hidden: boolean</td>
<td>True if the control is hidden. Inherited from Control.hidden</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>applyDesign</td>
<td>applyDesign(IDesign: PartDesign): void</td>
<td>Applies given design to the design on the control. Overrides Control.applyDesign</td>
</tr>
<tr>
<td>dataContext</td>
<td>dataContext(): any</td>
<td>Inherited from Control.dataContext</td>
</tr>
<tr>
<td>getControl</td>
<td>getControl(controlName: string): Control</td>
<td>Given the name of a control, returns the control instance. Inherited from ContainerControl.getControl</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>getControlById</td>
<td>getControlById(id: string): Control</td>
<td>Given the ID of a control, returns the control instance. Inherited from ContainerControl.getControlById</td>
</tr>
<tr>
<td>getDesign</td>
<td>getDesign(): Design</td>
<td>Returns the design object of this control. Inherited from Control.getDesign</td>
</tr>
<tr>
<td>getEntityRef</td>
<td>getEntityRef(): string</td>
<td>Gets value of entityRef binding to control.</td>
</tr>
<tr>
<td>getPartPage</td>
<td>getPartPage(): Page</td>
<td>Gets the page of the part.</td>
</tr>
<tr>
<td>hasTarget</td>
<td>hasTarget(): boolean</td>
<td>Returns true if the part has a target page.</td>
</tr>
<tr>
<td>isEditable</td>
<td>isEditable(): boolean</td>
<td>Boolean indicating if the control is editable. Inherited from Control.isEditable</td>
</tr>
<tr>
<td>metadata</td>
<td>metadata(): PartMetadata</td>
<td>Returns the metadata object of this control. Overrides ContainerControl.metadata</td>
</tr>
<tr>
<td>parent</td>
<td>parent(): Control</td>
<td>Page</td>
</tr>
<tr>
<td>root</td>
<td>root(): Page</td>
<td>Returns the root form instance (page) of this control. Inherited from Control.root</td>
</tr>
</tbody>
</table>

### PartDesign

**Hierarchy**

- ContainerControlDesign
  - PartDesign

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>alignItems</td>
<td>alignItems: string (optional)</td>
<td>This property is an alias for the CSS property &quot;align-items&quot;. Inherited from Design.alignItems</td>
</tr>
<tr>
<td>alignSelf</td>
<td>alignSelf: string (optional)</td>
<td>Inherited from Design.alignSelf</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>allowScroll</td>
<td>allowScroll: string (optional)</td>
<td>True if the container will allow scrolling when its items do not fit into the container's available space. If a container has an item which may scroll, then set this property to false to prevent nested scrolling areas. Inherited from ContainerControlDesign.allowScroll</td>
</tr>
<tr>
<td>background</td>
<td>background: string (optional)</td>
<td>The background color of the container. Inherited from ContainerControlDesign.background</td>
</tr>
<tr>
<td>bindings</td>
<td>bindings: any (optional)</td>
<td>Inherited from Design.bindings</td>
</tr>
<tr>
<td>border</td>
<td>border: &quot;none&quot;</td>
<td>&quot;solid&quot;</td>
</tr>
<tr>
<td>color</td>
<td>color: string (optional)</td>
<td>The foreground color of the container. Inherited from Design.color</td>
</tr>
<tr>
<td>design</td>
<td>design: PartDesign (optional)</td>
<td>Design for the target page.</td>
</tr>
<tr>
<td>flexFlow</td>
<td>flexFlow: string (optional)</td>
<td>Specifying this property makes the component a flex container component. Inherited from Design.flexFlow</td>
</tr>
<tr>
<td>flexSize</td>
<td>flexSize: string (optional)</td>
<td>One number or two numbers written as a string. For example, &quot;(size to grow) [(size-to-shrink)]&quot; to accommodate available space in the immediate flex container. Inherited from Design.flexSize</td>
</tr>
<tr>
<td>fontSize</td>
<td>fontSize: &quot;medium&quot;</td>
<td>&quot;xx-small&quot;</td>
</tr>
<tr>
<td>fontWeight</td>
<td>fontWeight: &quot;normal&quot;</td>
<td>&quot;bold&quot; (optional)</td>
</tr>
<tr>
<td>itemBorder</td>
<td>itemBorder: &quot;solid&quot;</td>
<td>&quot;none&quot; (optional)</td>
</tr>
<tr>
<td>items</td>
<td>items: string</td>
<td>Design [ ] (optional)</td>
</tr>
</tbody>
</table>
### justifyItems

**justyItems**: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Inherited from **Design.justifyItems**

### label

**label**: string (optional)

Inherited from **Design.label**

### labelPosition

**labelPosition**: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked. Inherited from **Design.labelPosition**

### name

**name**: string (optional)

Inherited from **Design.name**

### padding

**padding**: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. Inherited from **Design.padding**

### target

**target**: PageTarget (optional)

Target page of the part.

### type

**type**: ControlType (optional)

The type of the control as a string. Inherited from **Design.type**

---

**PartMetadata**

**Hierarchy**

- **ContainerControlMetadata**
  - **PartMetadata**

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoundEntity</td>
<td>BoundEntity: string (optional)</td>
<td>The entity to which the control is bound. Inherited from <strong>ControlMetadata.BoundEntity</strong></td>
</tr>
<tr>
<td>BoundField</td>
<td>BoundField: string (optional)</td>
<td>Inherited from <strong>ControlMetadata.BoundField</strong></td>
</tr>
<tr>
<td>Description</td>
<td>Description: string (optional)</td>
<td>Description of the control. Inherited from <strong>ControlMetadata.Description</strong></td>
</tr>
<tr>
<td>Editable</td>
<td>Editable: boolean (optional)</td>
<td>Boolean indicating if the control is editable. Inherited from <strong>ControlMetadataEditable</strong></td>
</tr>
<tr>
<td>ExtType</td>
<td>ExtType: ControlType (optional)</td>
<td>The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from <strong>ControlMetadata.ExtType</strong></td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HelpText</td>
<td>HelpText: string (optional)</td>
<td>The keyboard shortcut for a command. For example, “(Shift+F5)”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inherited from ControlMetadata.HelpText</td>
</tr>
<tr>
<td>Hidden</td>
<td>Hidden: boolean (optional)</td>
<td>Boolean indicating if the control is hidden or not.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inherited from ControlMetadata.Hidden</td>
</tr>
<tr>
<td>Id</td>
<td>Id: string (optional)</td>
<td>Identification string for a control.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inherited from ControlMetadata.Id</td>
</tr>
<tr>
<td>Label</td>
<td>Label: string (optional)</td>
<td>Label for a control. For example, a control representing a person's first name might have a label &quot;First Name&quot;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inherited from ControlMetadata.Label</td>
</tr>
<tr>
<td>Name</td>
<td>Name: string (optional)</td>
<td>Name of a control.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inherited from ControlMetadata.Name</td>
</tr>
<tr>
<td>Order</td>
<td>Order: number (optional)</td>
<td>Number indicating the order in which a control will appear on a page.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inherited from ControlMetadata.Order</td>
</tr>
<tr>
<td>Target</td>
<td>Target: PageTarget (optional)</td>
<td>Target page of the part.</td>
</tr>
<tr>
<td>Type</td>
<td>Type: ControlType (optional)</td>
<td>String indicating the control type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inherited from ControlMetadata.Type</td>
</tr>
</tbody>
</table>
Various services that are available to the application in client runtime.

Index

Types

- AsyncService
- CacheService
- DataService
- MetadataService
- PageData

Type aliases

- ExpressionOperator

Types

AsyncService

Hierarchy

AsyncService

Methods

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>all(...args: any[]): Promise &lt;any&gt;</td>
<td></td>
</tr>
<tr>
<td>defer</td>
<td>defer &lt;T&gt;(): Deferred &lt;T&gt;</td>
<td>Creates a deferred object which can be used to return a promise from event handlers (where applicable) and resolve reject them asynchronously.</td>
</tr>
</tbody>
</table>

CacheService

Hierarchy

CacheService

Methods

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>getData</td>
<td>getData(cacheKey: string): any</td>
<td></td>
</tr>
<tr>
<td>setData</td>
<td>setData(cacheKey: string, data: any): any</td>
<td></td>
</tr>
</tbody>
</table>

DataService

Hierarchy

DataService

Methods
<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>findEntityData</td>
<td>findEntityData(entityType: any, propertyName: string, propertyValue: any, includeChanges?: boolean): any</td>
<td></td>
</tr>
<tr>
<td>getEntityData</td>
<td>getEntityData(entityType: any, entityId: string): any</td>
<td></td>
</tr>
<tr>
<td>getPageData</td>
<td>getPageData(pageld: string, context: any, filter: any, allowedStaleness: number): Promise &lt;PageData&gt;</td>
<td></td>
</tr>
</tbody>
</table>

**MetadataService**

**Hierarchy**

MetadataService

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>version: string</td>
<td>Gets the version of the platform currently running.</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>addControl</td>
<td>addControl(componentName: string, controlName: string, controlType: ControlType, parentContainerName?: string, options?: ControlMetadata): any</td>
<td></td>
</tr>
<tr>
<td>configureAction</td>
<td>configureAction(actionName: string, options: PageMetadata): any</td>
<td>Configuring an action allows specifying or overriding certain behaviors specific to actions.</td>
</tr>
<tr>
<td>configureControl</td>
<td>configureControl(componentName: string, controlName: string, options: ControlMetadata): any</td>
<td>Configuring a control allows specifying or overriding certain behaviors specific to the control. Note that the available behaviors vary by control type.</td>
</tr>
<tr>
<td>configureEntity</td>
<td>configureEntity(entityName: string, options: any): any</td>
<td>Configuring an entity allows specifying or overriding certain behaviors specific to the entity.</td>
</tr>
<tr>
<td>configureLookup</td>
<td>configureLookup(taskName: string, lookupControlName: string, options: LookupMetadata): any</td>
<td>Configures a field on an action to behave as a lookup. Requires using an existing page which contains a list control.</td>
</tr>
<tr>
<td>configurePage</td>
<td>configurePage(pageName: string, options: PageMetadata): any</td>
<td>Configuring a Page allows specifying or overriding certain behaviors specific to the Page.</td>
</tr>
</tbody>
</table>
### configureWorkspace

configureWorkspace(options: PageMetadata): any

Configuring a workspace allows specifying or overriding certain behaviors specific to the workspace.

### findAction

findAction(actionName: string): PageMetadata

Gets a copy of the current metadata instance of a specified Action, for the purpose of inspecting the metadata (not to be used for changing the metadata).

### findControl

findControl(componentMetadata: any, controlName: string): ControlMetadata

Gets a copy of the current metadata instance of a specified control, for the purpose of inspecting the metadata (not to be used for changing the metadata).

### findPage

findPage(pageName: string): PageMetadata

Gets a copy of the current metadata instance of a specified page, for the purpose of inspecting the metadata (not to be used for changing the metadata).

### getFilterExpression

getFilterExpression(pageName: string, listControlName: string, controlName: string, operator: ExpressionOperator, value: string): DataFilter

Create a DataFilter object for a list control based on the provided options.

### getFormReference

getFormReference(componentName: string, filterContext: DataFilter, excludeContext: boolean, filterLocalOnly?: boolean): NavigationArgs

Create an INavigationArgs object for a specific page action to be used with a navigation control.

### hideNavigation

hideNavigation(pageNamesToHide: string[]): any

Hides the specified page(s) from the default landing page.

### PageData

#### Hierarchy

PageData

#### Methods

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>getControlValue</td>
<td>getControlValue(controlName: string): any</td>
<td>Gets the value of a control directly from the data set loaded in the page.</td>
</tr>
<tr>
<td>setControlValue</td>
<td>setControlValue(controlName: string, value: any): any</td>
<td>Sets the value of a control directly into the data set loaded in the page.</td>
</tr>
</tbody>
</table>

### Type aliases

**ExpressionOperator**

ExpressionOperator: "Is" | "IsNot" | "Contains" | "BeginsWith" | "EndsWith" | "GreaterThan" | "LessThan" | "GreaterThanOrEqual" | "LessThanOrEqual"
Represents possible values for the expression operator used in defining filters and in other places
This is the base class for single value controls.

**Index**

**Types**
- GenericValue
- Value
- ValueDesign
- ValueMetadata

**Types**

**GenericValue**

**Hierarchy**

Value  
└— GenericValue

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>container: boolean (optional)</td>
<td>True if the control is a container. Inherited from Control.container</td>
</tr>
<tr>
<td>generic</td>
<td>generic: boolean</td>
<td>True if the control is a generic. Overrides Control.generic</td>
</tr>
<tr>
<td>getDataSource</td>
<td>getDataSource: function(): any</td>
<td>Inherited from Control.getDataSource</td>
</tr>
<tr>
<td>hidden</td>
<td>hidden: boolean</td>
<td>True if the control is hidden. Inherited from Control.hidden</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>applyDesign</td>
<td>applyDesign(design: Design): void</td>
<td>Applies given design to the design on the control. Inherited from Control.applyDesign</td>
</tr>
<tr>
<td>dataSource</td>
<td>dataSource(): any</td>
<td>Inherited from Control.dataSource</td>
</tr>
<tr>
<td>getDesign</td>
<td>getDesign(): Design</td>
<td>Returns the design object of this control. Inherited from Control.getDesign</td>
</tr>
<tr>
<td>getValue</td>
<td>getValue(): string</td>
<td>Returns the value of the control. Inherited from Value.getValue</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>isEditable</td>
<td>isEditable(): boolean</td>
<td>Boolean indicating if the control is editable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inherited from Control.isEditable</td>
</tr>
<tr>
<td>metadata</td>
<td>metadata(): ValueMetadata</td>
<td>Returns the metadata object of this control.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inherited from Value.metadata</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Overrides InputControl.metadata</td>
</tr>
<tr>
<td>parent</td>
<td>parent(): Control</td>
<td>Page</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inherited from Control.parent</td>
</tr>
<tr>
<td>root</td>
<td>root(): Page</td>
<td>Returns the root form instance (page) of this control.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inherited from Control.root</td>
</tr>
<tr>
<td>setValue</td>
<td>setValue(value: string): void</td>
<td>Sets the value of the control.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inherited from Value.setValue</td>
</tr>
</tbody>
</table>

**Events**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>onDataChange</td>
<td>onDataChange: EventHook &lt;null&gt;</td>
<td>An event that is triggered when the input control's data changes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inherited from InputControl.onDataChanged</td>
</tr>
</tbody>
</table>

**Value**

**Hierarchy**

InputControl
- Value
  - FileUploader
  - HyperLink
  - GenericValue

**Properties**

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>container</td>
<td>container: boolean (optional)</td>
<td>True if the control is a container.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inherited from Control.container</td>
</tr>
<tr>
<td>generic</td>
<td>generic: boolean (optional)</td>
<td>Inherited from Control.generic</td>
</tr>
<tr>
<td>getDataSource</td>
<td>getDataSource: function(): any</td>
<td>Inherited from Control.getDataSource</td>
</tr>
<tr>
<td>hidden</td>
<td>hidden: boolean</td>
<td>True if the control is hidden.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inherited from Control.hidden</td>
</tr>
</tbody>
</table>

**Methods**
applyDesign

applyDesign(design: Design): void
Applies given design to the design on the control. Inherited from Control.applyDesign

dataContext
dataContext(): any
Inherited from Control.dataContext

getDesign
getDesign(): Design
Returns the design object of this control. Inherited from Control.getDesign

getValue
getValue(): string
Returns the value of the control.

isEditable
isEditable(): boolean
Boolean indicating if the control is editable. Inherited from Control.isEditable

metadata
metadata(): ValueMetadata
Returns the metadata object of this control. Overrides InputControl.metadata

parent
parent(): Control | Page
Returns the parent (control or page) of this control. Inherited from Control.parent

root
root(): Page
Returns the root form instance (page) of this control. Inherited from Control.root

setValue
setValue(value: string): void
Sets the value of the control.

Events

onDataChanged
onDataChanged: EventHook <null>
An event that is triggered when the input control's data changes. Inherited from InputControl.onDataChanged

ValueDesign

Hierarchy

InputControlDesign
   └ ValueDesign
      └ FileUploaderDesign
      └ HyperLinkDesign

Properties

alignItems
alignItems: string (optional)
This property is an alias for the CSS property "align-items". Inherited from Design.alignItems

alignSelf
alignSelf: string (optional)
Inherited from Design.alignSelf
<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>bindings</td>
<td>bindings: any (optional)</td>
<td>Inherited from Design.bindings</td>
</tr>
<tr>
<td>border</td>
<td>border: &quot;none&quot;</td>
<td>&quot;solid&quot;</td>
</tr>
<tr>
<td>color</td>
<td>color: string (optional)</td>
<td>The foreground color of the container. Inherited from Design.color</td>
</tr>
<tr>
<td>flexFlow</td>
<td>flexFlow: string (optional)</td>
<td>Specifying this property makes the component a flex container component. Inherited from Design.flexFlow</td>
</tr>
<tr>
<td>flexSize</td>
<td>flexSize: string (optional)</td>
<td>One number or two numbers written as a string. For example, &quot;(size to grow) [(size-to-shrink)]&quot; to accommodate available space in the immediate flex container. Inherited from Design.flexSize</td>
</tr>
<tr>
<td>fontSize</td>
<td>fontSize: &quot;medium&quot;</td>
<td>&quot;xx-small&quot;</td>
</tr>
<tr>
<td>fontWeight</td>
<td>fontWeight: &quot;normal&quot;</td>
<td>&quot;bold&quot; (optional)</td>
</tr>
<tr>
<td>justifyItems</td>
<td>justifyItems: &quot;flex-start&quot;</td>
<td>&quot;flex-end&quot;</td>
</tr>
<tr>
<td>label</td>
<td>label: string (optional)</td>
<td>Inherited from Design.label</td>
</tr>
<tr>
<td>labelPosition</td>
<td>labelPosition: &quot;stacked&quot;</td>
<td>&quot;hidden&quot;</td>
</tr>
<tr>
<td>name</td>
<td>name: string (optional)</td>
<td>Inherited from Design.name</td>
</tr>
<tr>
<td>padding</td>
<td>padding: &quot;none&quot;</td>
<td>&quot;small&quot;</td>
</tr>
<tr>
<td>type</td>
<td>type: ControlType (optional)</td>
<td>The type of the control as a string. Inherited from Design.type</td>
</tr>
</tbody>
</table>

ValueMetadata
Hierarchy
InputControlMetadata
└── ValueMetadata
### Properties

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BoundEntity</strong></td>
<td><strong>BoundEntity: string (optional)</strong></td>
<td>The entity to which the control is bound. Inherited from ControlMetadata.BoundEntity</td>
</tr>
<tr>
<td><strong>BoundField</strong></td>
<td><strong>BoundField: string (optional)</strong></td>
<td>Inherited from ControlMetadata.BoundField</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td><strong>Description: string (optional)</strong></td>
<td>Description of the control. Inherited from ControlMetadata.Description</td>
</tr>
<tr>
<td><strong>Editable</strong></td>
<td><strong>Editable: boolean (optional)</strong></td>
<td>Boolean indicating if the control is editable. Inherited from ControlMetadata.Editable</td>
</tr>
<tr>
<td><strong>ExtType</strong></td>
<td><strong>ExtType: ControlType (optional)</strong></td>
<td>The extended control type. For example, a control of type Input might have an extended type of Barcode. Inherited from ControlMetadata.ExtType</td>
</tr>
<tr>
<td><strong>HelpText</strong></td>
<td><strong>HelpText: string (optional)</strong></td>
<td>The keyboard shortcut for a command. For example, &quot;(Shift+F5)&quot; Inherited from ControlMetadata.HelpText</td>
</tr>
<tr>
<td><strong>Hidden</strong></td>
<td><strong>Hidden: boolean (optional)</strong></td>
<td>Boolean indicating if the control is hidden or not. Inherited from ControlMetadata.Hidden</td>
</tr>
<tr>
<td><strong>Id</strong></td>
<td><strong>Id: string (optional)</strong></td>
<td>Identification string for a control. Inherited from ControlMetadata.Id</td>
</tr>
<tr>
<td><strong>Label</strong></td>
<td><strong>Label: string (optional)</strong></td>
<td>Label for a control. For example, a control representing a person's first name might have a label &quot;First Name&quot;. Inherited from ControlMetadata.Label</td>
</tr>
<tr>
<td><strong>Mandatory</strong></td>
<td><strong>Mandatory: boolean (optional)</strong></td>
<td>If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline. Inherited from InputControlMetadata.Mandatory</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td><strong>Name: string (optional)</strong></td>
<td>Name of a control. Inherited from ControlMetadata.Name</td>
</tr>
<tr>
<td>NAME</td>
<td>SIGNATURE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NumSequence</td>
<td>NumSequence: NumberSequenceConfig (optional)</td>
<td>Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Inherited from InputControlMetadata.NumSequence</td>
</tr>
<tr>
<td>Order</td>
<td>Order: number (optional)</td>
<td>Number indicating the order in which a control will appear on a page. Inherited from ControlMetadata.Order</td>
</tr>
<tr>
<td>Type</td>
<td>Type: ControlType (optional)</td>
<td>String indicating the control type. Inherited from ControlMetadata.Type</td>
</tr>
</tbody>
</table>
Represents a runtime instance of an application.

Hierarchy
Application

Index
Properties
- minVersion

Methods
- appInit

Properties

minVersion

minVersion: string (optional)

An optional marker to indicate the minimum platform version required by this component. When this is specified and the component is attempted to be loaded in an older version of the platform, the corresponding workspace is not loaded and user is directed to install a newer version of the platform.

Methods

appInit

appInit(metadata: ApplicationMetadata): any

This method is invoked at the point the application is about to run, with the instance of the application metadata loaded. The metadata passed in can be still modified to change behaviors before this method returns.

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>metadata</td>
<td>ApplicationMetadata</td>
<td></td>
</tr>
</tbody>
</table>

Returns any
Represents the declarative metadata of an application

**Hierarchy**
ApplicationMetadata

**Index**

**Properties**
- **ColorName**
  - ColorName: string (optional)
  - The theme color of the application
- **Configs**
  - Configs: {name: string}: any (optional)
  - An application can have a set of named config supplied by the author or the resource provider
- **Description**
  - Description: string (optional)
  - The description of the application
- **IconName**
  - IconName: string (optional)
  - The representative icon of the application
- **Id**
  - Id: string
  - The unique identifier of the application
- **Title**
  - Title: string
  - The title of the application
AsyncService type

Provides ability to perform async operations from business logic code.

**Hierarchy**
AsyncService

**Index**

**Method list**
- all
- defer

**Methods**

**all**

```typescript
all(...args: any[]): Promise<any[]>
```

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>...args</td>
<td>any[]</td>
<td></td>
</tr>
</tbody>
</table>

**Returns** Promise <any[]>

**defer**

```typescript
defer(): Deferred<T>
```

Creates a deferred object that can be used to return a promise from event handlers (where applicable) and resolve/reject them asynchronously.

**Returns** Deferred <T>
Provides ability to access data from the device cache and update data into the device cache.

Hierarchy
CacheService

Index
Methods
- `getData`
- `setData`

Methods

**getData**
`getData(cacheKey: string): any`

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>cacheKey</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

Returns any

**setData**
`setData(cacheKey: string, data: any): any`

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>cacheKey</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>data</td>
<td>any</td>
<td></td>
</tr>
</tbody>
</table>

Returns any
CompleteEventArgs type

Hierarchy
CompleteEventArgs

Index

Properties

- error
- navigation
- processed

Properties

error
error: boolean (optional)

navigation
navigation: NavigationArgs (optional)

processed
processed: boolean (optional)
Container control interface with methods and attributes for all container controls. A container control can contain any number of controls.

Hierarchy

Control
└ ContainerControl
  └ Group
  └ List
  └ Part

Index

Properties

• container
• generic
• getDataSource
• hidden

Methods

• applyDesign
• dataContext
• getControl
• getControlById
• getDesign
• isEditable
• metadata
• parent
• root

Properties

container
container: boolean

True if the control is a container.

Overrides Control.container

generic
generic: boolean (optional)

Inherited from Control.generic

getDataSource
gDataSource: function(): any
Inherited from `Control.getDataSource`

**hidden**

hidden: boolean

True if the control is hidden.

Inherited from `Control.hidden`

## Methods

**applyDesign**

`applyDesign(design: Design): void`

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

Inherited from `Control.applyDesign`

### Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>design</td>
<td>Design</td>
<td>object containing design properties as keys</td>
</tr>
</tbody>
</table>

Returns void

**dataContext**

dataContext(): any

Inherited from `Control.dataContext`

Returns any

**getControl**

getControl(controlName: string): `Control`

Given the name of a control, returns the control instance.

### Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlName</td>
<td>string</td>
<td>control name</td>
</tr>
</tbody>
</table>

Returns `Control`

**getControlById**

getControlById(id: string): `Control`

Given the ID of a control, returns the control instance.

### Parameters
<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>control ID</td>
</tr>
</tbody>
</table>

Returns *Control*

**getDesign**

getDesign(): *Design*

Returns the design object of this control.

Inherited from *Control.getDesign*

Returns *Design*

**isEditable**

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control’s edit-ability and its parent’s edit-ability is undefined.

Inherited from *Control.isEditable*

Returns boolean

**metadata**

metadata(): *ContainerControlMetadata*

Returns the metadata object of this control.

Overrides *Control.metadata*

Returns *ContainerControlMetadata*

**parent**

parent(): *Control | Page*

Returns the parent (control or page) of this control.

Inherited from *Control.parent*

Returns *Control | Page*

**root**

root(): *Page*

Returns the root form instance (page) of this control.

Inherited from *Control.root*

Returns *Page*
Container control design object has properties specific to all container controls.

Hierarchy

- Design
  - ContainerControlDesign
    - GroupDesign
    - ListDesign
    - PartDesign

Index

Properties

- alignItems
- alignSelf
- allowScroll
- background
- bindings
- border
- color
- flexFlow
- flexSize
- fontSize
- fontWeight
- itemBorderStyle
- items
- justifyItems
- label
- labelPosition
- name
- padding
- type

Properties

**alignItems**

alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to this web page for documentation on the "align-items" property.

Inherited from **Design.alignItems**

**alignSelf**

alignSelf: string (optional)
allowScroll

allowScroll: string (optional)

True if the container will allow scrolling when its items do not fit into the container's available space. If a container has an item which may scroll, then set this property to false to prevent nested scrolling areas.

background

background: string (optional)

The background color of the container. Consider modifying the color attribute in the same container so that fonts overlaying the background color will appear appropriately. Note: if background is set to "theme", the theme color of the app will be used. The following colors are available:

- blue: \#0078D7;
- pomegranate: \#911844;
- raspberry: \#8D398F;
- darkOrange: \#D24726;
- green: \#369F47;
- blueberry: \#17234E;
- grape: \#432158;
- lightBlue: \#5DB2FF;
- lightGreen: \#82BA00;
- pink: \#DC4FAD;
- teal: \#008299;
- mediumDarkBlue: \#004B8B;
- cordovan: \#570000;
- darkCordovan: \#380000;
- black: \#000000;
- lightGray: \#e8e8e8;
- light: \#fff;
- dark: \#333333;

bindings

bindings: any (optional)

border

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

color

color: string (optional)
The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.

Consider setting the background color at the same time as necessary when setting this attribute.

Note: if color is set to "theme", the theme color of the app will be used.

The following colors are available:

- blue: \#0078D7
- pomegranate: \#911844
- raspberry: \#8D398F
- darkOrange: \#D24726
- green: \#369F47
- blueberry: \#17234E
- grape: \#432158
- lightBlue: \#5DB2FF
- lightGreen: \#82BA00
- pink: \#DC4FAD
- teal: \#008299
- mediumDarkBlue: \#004B8B
- cordovan: \#570000
- darkCordovan: \#380000
- black: \#000000
- lightGray: \#e8e8e8
- light: \#fff
- dark: \#333333

Inherited from Design.color

flexFlow

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property “flex-flow”. Please refer to this web page for documentation on the “flex-flow” property.

Inherited from Design.flexFlow

flexSize

flexSize: string (optional)

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the “flex” property.

Inherited from Design.flexSize

fontSize

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size
Inherited from Design.fontSize

fontWeight
fontWeight: "normal" | "bold" (optional)
Normal or bold text.
Inherited from Design.fontWeight

itemBorder
itemBorder: "solid" | "none" (optional)
If true, a border will appear around each row in the list. This property is equivalent to applying the border property individually to all items in the container.

items
items: string | Design [] (optional)
An array containing the components to place inside of the container.

justifyItems
justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)
This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.
Inherited from Design.justifyItems

label
label: string (optional)
Inherited from Design.label

labelPosition
labelPosition: "stacked" | "hidden" | "inline" (optional)
Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.
Inherited from Design.labelPosition

name
name: string (optional)
Inherited from Design.name

padding
padding: "none" | "small" | "std" (optional)
Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.
Inherited from Design.padding

type
type: `ControlType` (optional)

The type of the control as a string.

Inherited from `Design.type`
Container control metadata type.

Hierarchy

ControlMetadata
  ├ ContainerControlMetadata
  │   └ GroupMetadata
  │   └ ListMetadata
  │   └ PartMetadata

Index

Properties

- **BoundEntity**
  The entity to which the control is bound.

- **BoundField**

- **Description**
  Description of the control.

- **Editable**

- **ExtType**

- **HelpText**

- **Hidden**

- **Id**

- **Label**

- **Name**

- **Order**

- **Type**

Properties

**BoundEntity**

BoundEntity: string (optional)

The entity to which the control is bound.

Inherited from **ControlMetadata.BoundEntity**

**BoundField**

BoundField: string (optional)

Inherited from **ControlMetadata.BoundField**

**Description**

Description: string (optional)

Description of the control.

Inherited from **ControlMetadata.Description**
**Editable**

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

Inherited from `ControlMetadataEditable`

**ExtType**

ExtType: `ControlType` (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

Inherited from `ControlMetadataExtType`

**HelpText**

HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

Inherited from `ControlMetadataHelpText`

**Hidden**

Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

Inherited from `ControlMetadataHidden`

**Id**

Id: string (optional)

Identification string for a control.

Inherited from `ControlMetadataId`

**Label**

Label: string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

Inherited from `ControlMetadataLabel`

**Name**

Name: string (optional)

Name of a control.

Inherited from `ControlMetadataName`

**Order**

Order: number (optional)
Number indicating the order in which a control will appear on a page.

Inherited from ControlMetadata.Order

**Type**

Type: ControlType (optional)

String indicating the control type.

Inherited from ControlMetadata.Type
Interface for the metadata of a control. Overriding control metadata can modify a controls' look and behavior. Properties that can be modified vary by control but every control will have the base properties listed here.

**Hierarchy**

ControlMetadata
  └ PageLinkMetadata
  └ ContainerControlMetadata
  └ InputControlMetadata
  └ ImageMetadata

**Index**

**Properties**

- **BoundEntity**
- **BoundField**
- **Description**
- **Editable**
- **ExtType**
- **HelpText**
- **Hidden**
- **Id**
- **Label**
- **Name**
- **Order**
- **Type**

**Properties**

**BoundEntity**

BoundEntity: string (optional)

The entity to which the control is bound.

**BoundField**

BoundField: string (optional)

**Description**

Description: string (optional)

Description of the control.

**Editable**

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is
undefined. Undefined if both the control’s edit-ability and its parent’s edit-ability is undefined.

**ExtType**
ExtType: ControlType (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

**HelpText**
HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

**Hidden**
Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

**Id**
Id: string (optional)

Identification string for a control.

**Label**
Label: string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

**Name**
Name: string (optional)

Name of a control.

**Order**
Order: number (optional)

Number indicating the order in which a control will appear on a page.

**Type**
Type: ControlType (optional)

String indicating the control type.
DataService type

11/24/2021 • 2 minutes to read • Edit Online

Provides ability access data under the application workspace.

Hierarchy
DataService

Index

Methods

- **findEntityData**
- **getEntityData**
- **getPageData**

### Methods

#### findEntityData

```
findEntityData(entityType: any, propertyName: string, propertyValue: any, includeChanges?: boolean): any
```

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>entityType</td>
<td>any</td>
<td></td>
</tr>
<tr>
<td>propertyName</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>propertyValue</td>
<td>any</td>
<td></td>
</tr>
<tr>
<td>includeChanges?</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

any

#### getEntityData

```
getEntityData(entityType: any, entityId: string): any
```

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>entityType</td>
<td>any</td>
<td></td>
</tr>
<tr>
<td>entityId</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

any

#### getPageData

```
getPageData(pageId: string, context: any, filter: any, allowedStaleness: number): Promise <PageData>
```

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>pageId</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>context</td>
<td>any</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>any</td>
<td></td>
</tr>
<tr>
<td>allowedStaleness</td>
<td>number</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

Promise <PageData>
<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>pageld</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>context</td>
<td>any</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>any</td>
<td></td>
</tr>
<tr>
<td>allowedStaleness</td>
<td>number</td>
<td></td>
</tr>
</tbody>
</table>

Returns Promise `<PageData>"
Deferred type<T>

Hierarchy
Deferred

Index

Properties
- promise

Methods
- reject
- resolve

Properties

promise
promise: Promise <T>

Methods

reject
reject(error?: any): void

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>error?</td>
<td>any</td>
<td></td>
</tr>
</tbody>
</table>

Returns void

resolve
resolve(value?: T | PromiseLike <T>): void

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>value?</td>
<td>T</td>
<td>PromiseLike &lt;T&gt;</td>
</tr>
</tbody>
</table>

Returns void
Design object type. For more information on design, please reference the Design Introduction.

**Hierarchy**

Design
└── PageLinkDesign
  └── ContainerControlDesign
  └── InputControlDesign
  └── ImageDesign

**Index**

**Properties**

- alignItems
- alignSelf
- bindings
- border
- color
- flexFlow
- flexSize
- fontSize
- fontWeight
- justifyItems
- label
- labelPosition
- name
- padding
- type

**Properties**

**alignItems**

alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to this web page for documentation on the "align-items" property.

**alignSelf**

alignSelf: string (optional)

**bindings**

bindings: any (optional)

**border**

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)
The border behavior of a control. This property will not be inherited by the children.

**color**
color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.
Consider setting the background color at the same time as necessary when setting this attribute.
Note: if color is set to “theme”, the theme color of the app will be used.

The following colors are available:

- **blue**: #0078D7;
- **pomegranate**: #911B44;
- **raspberry**: #8D398F;
- **darkOrange**: #D24726;
- **green**: #369F47;
- **blueberry**: #17234E;
- **grape**: #432158;
- **lightBlue**: #5DB2FF;
- **lightGreen**: #82BA00;
- **pink**: #DC4FAD;
- **teal**: #008299;
- **mediumDarkBlue**: #004B8B;
- **cordovan**: #570000;
- **darkCordovan**: #380000;
- **black**: #000000;
- **lightGray**: #e8e8e8;
- **light**: #fff;
- **dark**: #333333;

**flexFlow**
flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property “flex-flow”. Please refer to this web page for documentation on the “flex-flow” property.

**flexSize**
flexSize: string (optional)

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the “flex” property.

**fontSize**
fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

**fontWeight**
fontWeight: "normal" | "bold" (optional)
Normal or bold text.

**justifyItems**
justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

**label**
label: string (optional)

**labelPosition**
labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

**name**
name: string (optional)

**padding**
padding: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

**type**
type: ControlType (optional)

The type of the control as a string.
EventHook type<T>

11/24/2021 • 2 minutes to read • Edit Online

Hierarchy
EventHook

Index

Methods

• subscribe
• unsubscribe
• unsubscribeAll

Methods

subscribe
subscribe(listener: IEventListener<T>): void

Subscribe a listener to this event.

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>listener</td>
<td>IEventListener&lt;T&gt;</td>
<td></td>
</tr>
</tbody>
</table>

Returns void

unsubscribe
unsubscribe(listener: IEventListener<T>): void

Unsubscribe a listener from this event.

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>listener</td>
<td>IEventListener&lt;T&gt;</td>
<td></td>
</tr>
</tbody>
</table>

Returns void

unsubscribeAll
unsubscribeAll(): void

Remove all listeners from this event.

Returns void
Field control type.

**Hierarchy**

InputControl
  └ Field

**Index**

**Properties**
- container
- generic
- getDataSource
- hidden

**Methods**
- applyDesign
- DataContext
- getDesign
- getEditableFormattedValue
- getEditableValue
- getEntityRef
- getFormattedValue
- getRefLink
- getValue
- hasRefLink
- hasUnWrapText
- isEditable
- metadata
- parent
- root
- setEditableValue

**Events**
- onDataChanged

**Properties**

- **container**
  
  container: boolean (optional)

  True if the control is a container.

  *Inherited from Control.container*
**generic**
generic: boolean (optional)

Inherited from `Control.generic`

**getDataSource**

getDataSource: function(): any

Inherited from `Control.getDataSource`

**hidden**

hidden: boolean

True if the control is hidden.

Inherited from `Control.hidden`

## Methods

### applyDesign

**applyDesign**(IDesign: `FieldDesign`): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

Overrides `Control.applyDesign`

### Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDesign</td>
<td><code>FieldDesign</code></td>
<td>object containing design properties as keys</td>
</tr>
</tbody>
</table>

**Returns void**

### dataContext

**dataContext**(): any

Inherited from `Control.dataContext`

**Returns any**

### getDesign

**getDesign**(): `Design`

Returns the design object of this control.

Inherited from `Control.getDesign`

**Returns Design**

### getEditableFormattedValue

**getEditableFormattedValue**(): string | number | Date

Gets a formatted decimal string value of an editable field control.
getEditableValue

getEditableValue(): string | number | Date

Gets the value for an editable field control.

getEntityRef

getEntityRef(): any

Gets value of entityRef binding to control.

getFormattedValue

getFormattedValue(): string

Gets a formatted decimal string value.

getRefLink

getRefLink(): NavigationArgs

Gets the navigation object for a reference link.

getValue

getValue(): any

Gets the value for a field control.

hasRefLink

hasRefLink(): boolean

Returns true if the field has a refLink, otherwise false.

hasUnWrapText

hasUnWrapText(): boolean

Gets wrap text property of control.

isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control’s edit-ability and its parent’s edit-ability is undefined.

Inherited from Control.isEditable

Returns boolean
metadata

metadata(): FieldMetadata

Returns the metadata object of this control.

| Overides InputControl.metadata |

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

| Inherited from Control.parent |

root

root(): Page

Returns the root form instance (page) of this control.

| Inherited from Control.root |

setEditableValue

setEditableValue(value: string | number | Date): void

Sets the value for an editable field control.

| Parameters |
| NAME | TYPE | DESCRIPTION |
| value | string | number | Date | value |

Events

onDataChanged

onDataChanged: EventHook <null>

An event that is triggered when the input control's data changes.

| Inherited from InputControl.onDataChanged |
Design object interface for a field control.

**Hierarchy**

- InputControlDesign
  - FieldDesign

**Index**

**Properties**

- `alignItems`
- `alignSelf`
- `bindings`
- `border`
- `color`
- `flexFlow`
- `flexSize`
- `fontWeight`
- `justifyItems`
- `label`
- `labelPosition`
- `name`
- `padding`
- `type`

**Properties**

- **alignItems**
  
  `alignItems: string (optional)`

  This property is an alias for the CSS property "align-items". Please refer to [this web page](#) for documentation on the "align-items" property.

  - Inherited from Design.alignItems

- **alignSelf**
  
  `alignSelf: string (optional)`

  - Inherited from Design.alignSelf

- **bindings**
  
  `bindings: any (optional)`

  - Inherited from Design.bindings
**border**

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

Inherited from Design.border

**color**

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.

Consider setting the background color at the same time as necessary when setting this attribute.

Note: if color is set to "theme", the theme color of the app will be used.

The following colors are available:

- **blue**: #0078D7;
- **pomegranate**: #911844;
- **raspberry**: #8D398F;
- **darkOrange**: #D24726;
- **green**: #369F47;
- **blueberry**: #17234E;
- **grape**: #432158;
- **lightBlue**: #5DB2FF;
- **lightGreen**: #82BA00;
- **pink**: #DC4FAD;
- **teal**: #008299;
- **mediumDarkBlue**: #004B8B;
- **cordovan**: #570000;
- **darkCordovan**: #380000;
- **black**: #000000;
- **lightGray**: #e8e8e8;
- **light**: #fff;
- **dark**: #333333;

Inherited from Design.color

**flexFlow**

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

Inherited from Design.flexFlow

**flexSize**

flexSize: string (optional)
One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

Inherited from Design.flexSize

**fontSize**

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

Inherited from Design.fontSize

**fontWeight**

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

Inherited from Design.fontWeight

**justifyItems**

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

Inherited from Design.justifyItems

**label**

label: string (optional)

Inherited from Design.label

**labelPosition**

labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

Inherited from Design.labelPosition

**name**

name: string (optional)

Inherited from Design.name

**padding**

padding: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

Inherited from Design.padding
**type**

Type: `ControlType` (optional)

The type of the control as a string.

Inherited from `Design.type`
Interface for field metadata.

Hierarchy

InputControlMetadata
    └ FieldMetadata

Index

Properties

- BoundEntity
- BoundField
- DecimalPlaces
- Description
- Editable
- ExtType
- Formatting
- HelpText
- Hidden
- Id
- Label
- LinkType
- Mandatory
- Name
- NumSequence
- Order
- ReferenceAppId
- ReferencePageId
- Style
- Type
- UnWrapText
- WrapText

Properties

**BoundEntity**

BoundEntity: string (optional)

The entity to which the control is bound.

Inherited from ControlMetadata.BoundEntity
DecimalPlaces

DecimalPlaces: number (optional)

The number of decimals that appear on a field of type "Real". Default = 2; number must be in the range [0:20].

Description

Description: string (optional)

Description of the control.

Editable

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

ExtType

ExtType: ControlType (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

Formatting

Formatting: any (optional)

Formats a field of type "DateTime" or "Date". **Note**: if browser does not support `toLocaleString` with options then it will show the entire value.

The options for Formatting depends on the Style that has been chosen: Style: "DateOnly" options, Style: "TimeOnly" options, and options for no style.

Example 1: `{ Style: "TimeOnly", Formatting: { timeZone: "UTC", timeZoneName: "short" } }`

Example 2: `{ Style: "DateOnly", Formatting: { month: "long", day: "numeric" } }` result: March 2

HelpText

HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

Hidden

Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.
| Inherited from **ControlMetadata.Hidden** |

**Id**
Id: string (optional)
Identification string for a control.
| Inherited from **ControlMetadata.Id** |

**Label**
Label: string (optional)
Label for a control. For example, a control representing a person's first name might have a label "First Name".
| Inherited from **ControlMetadata.Label** |

**LinkType**
LinkType: "Telephone" | "Email" | "Url" (optional)
Assigning the link type of a field allows for the appropriate mobile application to be opened when the link is selected.

**Mandatory**
Mandatory: boolean (optional)
If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline.
| Inherited from **InputControlMetadata.Mandatory** |

**Name**
Name: string (optional)
Name of a control.
| Inherited from **ControlMetadata.Name** |

**NumSequence**
NumSequence: **NumberSequenceConfig** (optional)
Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Example:

```javascript
// hide number sequence reference page from users
metadataService.hideNavigation('numSeqReferencePage');

// parameters to be passed to 'numSequence' flag in configureControl
var configParam = {
    referencePageName: 'numSeqReferencePage',
    dataType: 'HcmPersonnelNumberId'
};

// setup 'PersonnelNumber' control as number sequence in the task 'add-worker'
metadataService.configureControl('add-worker', 'PersonnelNumber', { numSequence: configParam });
```
**Order**

Order: number (optional)

Number indicating the order in which a control will appear on a page.

Inherited from `ControlMetadata.Order`

**ReferenceAppId**

ReferenceAppId: string (optional)

The ID of the app that the field control lives in.

**ReferencePageId**

ReferencePageId: string (optional)

The ID of the page that the field control lives in.

**Style**

Style: string (optional)

Styles a field of type "DateTime" or "Date". Example: `{ Style: TimeOnly }` result: 12:00:00 AM

**Type**

Type: `ControlType` (optional)

String indicating the control type.

Inherited from `ControlMetadata.Type`

**UnWrapText**

UnWrapText: boolean (optional)

False by default -- text of the page will be wrapped.

**WrapText**

WrapText: boolean (optional)

If true then the text of the field control will wrap to the next line.
File uploader control type. A control for uploading files such as images.

**Hierarchy**
- **Value**
  - FileUploader

**Index**

**Properties**
- container
- generic
- getDataSource
- hidden
- image

**Methods**
- applyDesign
- canLoadFromDevice
- dataContext
- getDesign
- getImage
- getValue
- isEditable
- loadFromFileSystem
- metadata
- parent
- root
- setCamera
- setValue

**Events**
- onDataChange

**Properties**

**container**
container: boolean (optional)
True if the control is a container.

Inherited from Control.container

**generic**
generic: boolean (optional)
getDataSource

getDataSource: function(): any

Inherited from Control.getDataSource

hidden

hidden: boolean

True if the control is hidden.

Inherited from Control.hidden

image

image: Image

Methods

applyDesign

applyDesign(IDesign: FileUploaderDesign): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

Overrides Control.applyDesign

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDesign</td>
<td>FileUploaderDesign</td>
<td>object containing design properties as keys</td>
</tr>
</tbody>
</table>

Returns void
canLoadFromDevice
canLoadFromDevice(): boolean

Returns true if the mobile phone has camera plugin.

Returns boolean
dataContext
dataContext(): any

Inherited from Control.dataContext

Returns any
getDesign
getDesign(): Design

Returns the design object of this control.

Inherited from Control.getDesign
**getImage**

getImage(options: any): Promise <string>

Returns a promise of an object with image data.

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>options</td>
<td>any</td>
<td>see camera options</td>
</tr>
</tbody>
</table>

**getValue**

getValue(): any

Gets the value of the entity that is bound to the control.

Overrides Value.getValue

**isEditable**

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's editability and its parent's editability is undefined.

Inherited from Control.isEditable

**loadFromFileSystem**

loadFromFileSystem(file: Blob): Promise <any>

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>Blob</td>
<td></td>
</tr>
</tbody>
</table>

Returns Promise <any>

**metadata**

metadata(): FileUploaderMetadata

Returns the metadata object of this control.

Overrides Value.metadata

**parent**

parent(): Control | Page
Returns the parent (control or page) of this control.

<table>
<thead>
<tr>
<th>Returns</th>
<th>Control</th>
<th>Page</th>
</tr>
</thead>
</table>

**root**

root(): Page

Returns the root form instance (page) of this control.

<table>
<thead>
<tr>
<th>Returns</th>
<th>Control</th>
<th>Page</th>
</tr>
</thead>
</table>

**setCamera**

setCamera(camera: any): void

Set the camera object on the control.

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>camera</td>
<td>any</td>
<td></td>
</tr>
</tbody>
</table>

**Returns void**

**setValue**

setValue(value: string): void

Sets the value of the control.

<table>
<thead>
<tr>
<th>Returns</th>
<th>Control</th>
<th>Page</th>
</tr>
</thead>
</table>

**Events**

**onDataChanged**

onDataChanged: EventHook <null>

An event that is triggered when the input control's data changes.

<table>
<thead>
<tr>
<th>Returns</th>
<th>Control</th>
<th>Page</th>
</tr>
</thead>
</table>

Inherited from Control.parent

Inherited from Control.root

Inherited from Value.setValue

Inherited from InputControl.onDataChanged
File uploader design object type.

**Hierarchy**

ValueDesign

  ├── FileUploaderDesign

**Index**

**Properties**

- `alignItems`
- `alignSelf`
- `bindings`
- `border`
- `color`
- `flexFlow`
- `flexSize`
- `fontWeight`
- `justifyItems`
- `label`
- `labelPosition`
- `name`
- `padding`
- `type`

**Properties**

**alignItems**

`alignItems` is a string (optional)

This property is an alias for the CSS property "align-items". Please refer to [this web page](#) for documentation on the "align-items" property.

Inherited from Design.alignItems

**alignSelf**

`alignSelf` is a string (optional)

Inherited from Design.alignSelf

**bindings**

`bindings` is any (optional)

Inherited from Design.bindings
border
border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

Inherited from Design.border

color
color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.
Consider setting the background color at the same time as necessary when setting this attribute.
Note: if color is set to "theme", the theme color of the app will be used.

The following colors are available:

- blue: #0078D7
- pomegranate: #911844
- raspberry: #8D398F
- darkOrange: #D24726
- green: #369F47
- blueberry: #17234E
- grape: #432158
- lightBlue: #5DB2FF
- lightGreen: #82BA00
- pink: #DC4FAD
- teal: #008299
- mediumDarkBlue: #004B8B
- cordovan: #570000
- darkCordovan: #380000
- black: #000000
- lightGray: #e8e8e8
- light: #fff
- dark: #333333

Inherited from Design.color

flexFlow
flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

Inherited from Design.flexFlow

flexSize
flexSize: string (optional)
One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

| Inherited from Design.flexSize |

**fontSize**

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

| Inherited from Design.fontSize |

**fontWeight**

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

| Inherited from Design.fontSize |

**justifyItems**

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

| Inherited from Design.justifyItems |

**label**

label: string (optional)

| Inherited from Design.label |

**labelPosition**

labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

| Inherited from Design.labelPosition |

**name**

name: string (optional)

| Inherited from Design.name |

**padding**

padding: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

| Inherited from Design.padding |
**type**

*type: ControlType (optional)*

The type of the control as a string.

Inherited from `Design.type`
File uploader metadata type.

**Hierarchy**

ValueMetadata
  └ FileUploaderMetadata

**Index**

**Properties**

- **BoundEntity**
  BoundEntity: string (optional)

  The entity to which the control is bound.

  Inherited from ControlMetadata.BoundEntity

- **BoundField**
  BoundField: string (optional)

  Inherited from ControlMetadata.BoundField

- **Description**
  Description: string (optional)

  Description of the control.

  Inherited from ControlMetadata.Description
Editable
Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

Inherited from ControlMetadataEditable

ExtType
ExtType: ControlType (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

Inherited from ControlMetadataExtType

HelpText
HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

Inherited from ControlMetadataHelpText

Hidden
Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

Inherited from ControlMetadataHidden

Id
Id: string (optional)

Identification string for a control.

Inherited from ControlMetadataId

Label
Label: string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

Inherited from ControlMetadataLabel

Mandatory
Mandatory: boolean (optional)

If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline.

Inherited from InputControlMetadataMandatory

Name
Name: string (optional)
NumSequence

NumSequence: NumberSequenceConfig (optional)

Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Example:

```javascript
// hide number sequence reference page from users
metadataService.hideNavigation('numSeqReferencePage');

// parameters to be passed to 'numSequence' flag in configureControl
var configParam = {
    referencePageName: 'numSeqReferencePage',
    dataType: 'HcmPersonnelNumberId'
};

// setup 'PersonnelNumber' control as number sequence in the task 'add-worker'
metadataService.configureControl('add-worker', 'PersonnelNumber', { numSequence: configParam });
```

Order

Order: number (optional)

Number indicating the order in which a control will appear on a page.

Type

Type: ControlType (optional)

String indicating the control type.
Generic value control type.

**Hierarchy**
- Value
  - GenericValue

**Index**

**Properties**
- container
- generic
- getDataSource
- hidden

**Methods**
- applyDesign
- dataContext
- getDesign
- getValue
- isEditable
- metadata
- parent
- root
- setValue

**Events**
- onDataChange

**Properties**

**container**
container: boolean (optional)

True if the control is a container.

Inherited from Control.container

**generic**
generic: boolean

True if the control is a generic.

Overrides Control.generic

**getDataSource**
**applyDesign**

applyDesign(design: Design): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

Returns void

**dataContext**

dataContext(): any

Returns any

**getDesign**

getDesign(): Design

Returns the design object of this control.

Returns Design

**getValue**

getValue(): string

Returns the value of the control.

Returns string

**isEditable**

isEditable(): boolean
Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

| Inherited from | Control.isEditable |

**Returns boolean**

**metadata**

metadata(): `ValueMetadata`

Returns the metadata object of this control.

| Inherited from | Value.metadata |
| Inherited from | InputControl.metadata |

**Returns ValueMetadata**

**parent**

parent(): `Control` | `Page`

Returns the parent (control or page) of this control.

| Inherited from | Control.parent |

**Returns Control` | `Page**

**root**

root(): `Page`

Returns the root form instance (page) of this control.

| Inherited from | Control.root |

**Returns Page**

**setValue**

setValue(value: string): `void`

Sets the value of the control.

| Inherited from | Value.setValue |

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

**Returns void**

**Events**

**onDataChanged**
onDataChanged: EventHook <null>

An event that is triggered when the input control's data changes.

Inherited from InputControl.onDataChanged
Control interface with base methods and attributes for all controls. This represents the runtime instance of a control. Modifying the properties are immediately reflected in the UI.

### Hierarchy

Control
- PageLink
- ContainerControl
- InputControl
- Image

### Index

#### Properties
- container
- generic
- getDataSource
- hidden

#### Methods
- applyDesign
- dataContext
- getDesign
- isEditable
- metadata
- parent
- root

### Properties

**container**

container: boolean (optional)

True if the control is a container.

**generic**

generic: boolean (optional)

**getDataSource**

gerDataSource: function(): any

**hidden**

hidden: boolean

True if the control is hidden.

### Methods
applyDesign
design: Design: void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>design</td>
<td>Design</td>
<td>object containing design properties as keys</td>
</tr>
</tbody>
</table>

Returns void
dataContext
dataContext(): any

Returns any
getDesign
getDesign(): Design

Returns the design object of this control.

Returns Design

isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control’s edit-ability and its parent’s edit-ability is undefined.

Returns boolean

metadata

metadata(): ControlMetadata

Returns the metadata object of this control.

Returns ControlMetadata

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

Returns Control | Page

root

root(): Page

Returns the root form instance (page) of this control.

Returns Page
Group container control type. A group control is a container control that has any number of controls as children.

Hierarchy
ContainerControl
  └ Group

Index

Properties
- container
- generic
- getDataSource
- hidden

Methods
- applyDesign
- dataContext
- getChildren
- getControl
- getControlById
- getDesign
- isEditable
- metadata
- parent
- root

Properties

container
container: boolean

True if the control is a container.

<table>
<thead>
<tr>
<th>Inherited from</th>
<th>ContainerControl.container</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overrides</td>
<td>Control.container</td>
</tr>
</tbody>
</table>

generic
generic: boolean (optional)

| Inherited from | Control.generic |

groupDataSource
groupDataSource: function(): any
hidden
hidden: boolean
True if the control is hidden.

Methods

**applyDesign**

applyDesign(IDesign: GroupDesign): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDesign</td>
<td>GroupDesign</td>
<td>object containing design properties as keys</td>
</tr>
</tbody>
</table>

Returns void

**dataContext**

dataContext(): any

Returns any

**getChildren**

getchildren(): Control []

Returns the list of children associated with this group control.

**getControl**

getchildren(controlName: string): Control

Given the name of a control, returns the control instance.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlName</td>
<td>string</td>
<td>control name</td>
</tr>
</tbody>
</table>

Returns Control
getControlById(id: string): Control

Given the ID of a control, returns the control instance.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>control ID</td>
</tr>
</tbody>
</table>

Returns Control
getchildren

getchildren(): Control

Returns the children of this control.

<table>
<thead>
<tr>
<th>Inherited from</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ContainerControl.getControlById</td>
<td></td>
</tr>
</tbody>
</table>

getDesign
getchildren

getchildren(): Design

Returns the design object of this control.

<table>
<thead>
<tr>
<th>Inherited from</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Control.getDesign</td>
<td></td>
</tr>
</tbody>
</table>

isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control’s edit-ability and its parent’s edit-ability is undefined.

<table>
<thead>
<tr>
<th>Inherited from</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Control.isEditable</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns boolean</th>
<th></th>
</tr>
</thead>
</table>

metadata

metadata(): GroupMetadata

Returns the metadata object of this control.

<table>
<thead>
<tr>
<th>Overrides</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ContainerControl.metadata</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns GroupMetadata</th>
<th></th>
</tr>
</thead>
</table>

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

<table>
<thead>
<tr>
<th>Inherited from</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Control.parent</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns Control</th>
<th>Page</th>
</tr>
</thead>
</table>

root

root(): Page

Returns the root form instance (page) of this control.
Inherited from `Control.root`

Returns `Page`
Group design object type.

**Hierarchy**

```
ContainerControlDesign
 └── GroupDesign
```

**Index**

**Properties**

- `alignItems`
- `alignSelf`
- `allowScroll`
- `background`
- `bindings`
- `border`
- `color`
- `flexFlow`
- `flexSize`
- `fontSize`
- `fontWeight`
- `itemBorder`
- `items`
- `justifyItems`
- `label`
- `labelPosition`
- `name`
- `padding`
- `type`

**Properties**

**alignItems**

`alignItems: string (optional)`

This property is an alias for the CSS property "align-items". Please refer to [this web page](#) for documentation on the "align-items" property.

Inherited from `Design.alignItems`

**alignSelf**

`alignSelf: string (optional)`

Inherited from `Design.alignSelf`
allowScroll
allowScroll: string (optional)

True if the container will allow scrolling when its items do not fit into the container’s available space. If a container has an item which may scroll, then set this property to false to prevent nested scrolling areas.

| Inherited from | ContainerControlDesign.allowScroll |

background
background: string (optional)

The background color of the container. Consider modifying the color attribute in the same container so that fonts overlaying the background color will appear appropriately. Note: if background is set to “theme”, the theme color of the app will be used. The following colors are available:

- blue: #0078D7;
- pomegranate: #911844;
- raspberry: #8D398F;
- darkOrange: #D24726;
- green: #369F47;
- blueberry: #17234E;
- grape: #432158;
- lightBlue: #5DB2FF;
- lightGreen: #82BA00;
- pink: #DC4FAD;
- teal: #008299;
- mediumDarkBlue: #04B8B;
- cordovan: #570000;
- darkCordovan: #380000;
- black: #000000;
- lightGray: #e8e8e8;
- light: #fff;
- dark: #333333;

| Inherited from | ContainerControlDesign.background |

bindings
bindings: any (optional)

| Inherited from | Design.bindings |

border
border: “none” | “solid” | “left” | “right” | “top” | “bottom” (optional)

The border behavior of a control. This property will not be inherited by the children.

| Inherited from | Design.border |
**color**

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.

Consider setting the background color at the same time as necessary when setting this attribute.

Note: if color is set to "theme", the theme color of the app will be used.

The following colors are available:

- **blue**: #0078D7
- **pomegranate**: #911844
- **raspberry**: #8D398F
- **darkOrange**: #D24726
- **green**: #369F47
- **blueberry**: #17234E
- **grape**: #432158
- **lightBlue**: #5DB2FF
- **lightGreen**: #82BA00
- **pink**: #DC4FAD
- **teal**: #088299
- **mediumDarkBlue**: #004B8B
- **cordovan**: #570000
- **darkCordovan**: #380000
- **black**: #000000
- **lightGray**: #e8e8e8
- **light**: #fff
- **dark**: #333333

Inherited from Design.color

**flexFlow**

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

Inherited from Design.flexFlow

**flexSize**

flexSize: string (optional)

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

Inherited from Design.flexSize

**fontSize**
The proportional text size

Inherited from Design.fontSize

**fontWeight**

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

Inherited from Design.fontWeight

**itemBorder**

itemBorder: "solid" | "none" (optional)

If true, a border will appear around each row in the list. This property is equivalent to applying the border property individually to all items in the container.

Inherited from ContainerControlDesign.itemBorder

**items**

items: string | Design [] (optional)

An array containing the components to place inside of the container.

Inherited from ContainerControlDesign.items

**justifyItems**

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

Inherited from Design.justifyItems

**label**

label: string (optional)

Inherited from Design.label

**labelPosition**

labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

Inherited from Design.labelPosition

**name**

name: string (optional)

Inherited from Design.name

**padding**
**padding**: "none" | "small" | "std" (optional)

Allows specifying the component’s padding behavior. A component will inherit the padding behavior specified by its parent container components.

Inherited from `Design.padding`

---

**type**

type: `ControlType` (optional)

The type of the control as a string.

Inherited from `Design.type`
Group metadata type.

Hierarchy

- ContainerControlMetadata
  - GroupMetadata

Index

Properties

- **BoundEntity**
  BoundEntity: string (optional)

  The entity to which the control is bound.

  Inherited from ControlMetadata.BoundEntity

- **BoundField**
  BoundField: string (optional)

  Inherited from ControlMetadata.BoundField

- **Children**
  Children: ControlMetadata[] (optional)

  List of control metadata for each child control.

- **Description**
  Description: string (optional)
Description of the control.

**Editable**

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control’s edit-ability and its parent’s edit-ability is undefined.

Inherited from `ControlMetadataEditable`

**ExtType**

ExtType: `ControlType` (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

Inherited from `ControlMetadataExtType`

**HelpText**

HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

Inherited from `ControlMetadataHelpText`

**Hidden**

Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

Inherited from `ControlMetadataHidden`

**Id**

Id: string (optional)

Identification string for a control.

Inherited from `ControlMetadataId`

**Label**

Label: string (optional)

Label for a control. For example, a control representing a person’s first name might have a label "First Name".

Inherited from `ControlMetadataLabel`

**Name**

Name: string (optional)

Name of a control.

Inherited from `ControlMetadataName`
### Order

**Order**: number (optional)

Number indicating the order in which a control will appear on a page.

Inherited from `ControlMetadata.Order`

### Type

**Type**: `ControlType` (optional)

String indicating the control type.

Inherited from `ControlMetadata.Type`
Hyperlink control type. Hyperlink control is a control to represent hyperlinks. Pagelinks can also be used in most cases.

Index

Properties
- container
- generic
- getDataSource
- hidden

Methods
- applyDesign
- dataContext
- getDesign
- getHyperLinkValue
- getValue
- isEditable
- isHyperLinkURLPresent
- metadata
- parent
- root
- setBaseURL
- setValue

Events
- onDataChanged

Properties

container
container: boolean (optional)

True if the control is a container.

Inherited from Control.container

generic
generic: boolean (optional)

Inherited from Control.generic
**getDataSource**
getDataSource: function(): any

Inherited from Control.getDataSource

**hidden**
hidden: boolean

True if the control is hidden.

Inherited from Control.hidden

**Methods**

**applyDesign**
applyDesign(IDesign: HyperLinkDesign): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

Overrides Control.applyDesign

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDesign</td>
<td>HyperLinkDesign</td>
<td>object containing design properties as keys</td>
</tr>
</tbody>
</table>

Returns void

**dataContext**
dataContext(): any

Inherited from Control.dataContext

Returns any

**getDesign**
getDesign(): Design

Returns the design object of this control.

Inherited from Control.getDesign

Returns Design

**getHyperLinkValue**
getHyperLinkValue(): string

Returns string

**getValue**
getValue(): string

Returns the value of the control.
Returns string

**isEditable**

**isEditable(): boolean**

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control’s editability and its parent’s editability is undefined.

Inherited from **Control.isEditable**

Returns boolean

**isHyperLinkURLPresent**

**isHyperLinkURLPresent(): boolean**

Returns boolean

**metadata**

**metadata(): HyperLinkMetadata**

Returns the metadata object of this control.

Overrides **Value.metadata**

Returns **HyperLinkMetadata**

**parent**

**parent(): Control | Page**

Returns the parent (control or page) of this control.

Inherited from **Control.parent**

Returns **Control | Page**

**root**

**root(): Page**

Returns the root form instance (page) of this control.

Inherited from **Control.root**

Returns **Page**

**setBaseURL**

**setBaseURL(url: string): any**

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

**Returns any**
**setValue**

`setValue(value: string): void`

Sets the value of the control.

Inherited from `Value.setValue`

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
</tr>
<tr>
<td>value</td>
</tr>
</tbody>
</table>

**Returns void**

**Events**

**onDataChanged**

`onDataChanged: EventHook <null>`

An event that is triggered when the input control's data changes.

Inherited from `InputControl.onDataChanged`
Hyperlink design object type.

Hierarchy

ValueDesign
  └ HyperLinkDesign

Index

Properties

- alignItems
- alignSelf
- bindings
- border
- color
- flexGrow
- flexSize
- fontWeight
- justifyItems
- label
- labelPosition
- name
- padding
- type

Properties

alignItems

alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to this web page for documentation on the "align-items" property.

| Inherited from Design.alignItems |

alignSelf

alignSelf: string (optional)

| Inherited from Design.alignSelf |

bindings

bindings: any (optional)

| Inherited from Design.bindings |
border

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

| Inherited from Design.border |

color

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.

Consider setting the background color at the same time as necessary when setting this attribute.

Note: if color is set to "theme", the theme color of the app will be used.

The following colors are available:

- **blue**: #0078D7
- **pomegranate**: #911844
- **raspberry**: #8D398F
- **darkOrange**: #D24726
- **green**: #369F47
- **blueberry**: #17234E
- **grape**: #432158
- **lightBlue**: #5DB2FF
- **lightGreen**: #82BA00
- **pink**: #DC4FAD
- **teal**: #008299
- **mediumDarkBlue**: #004B8B
- **cordovan**: #570000
- **darkCordovan**: #380000
- **black**: #000000
- **lightGray**: #E8E8E8
- **light**: #FFF
- **dark**: #333333

| Inherited from Design.color |

flexFlow

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

| Inherited from Design.flexFlow |

flexSize

flexSize: string (optional)
One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

- **fontSize**
  
  fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)
  
  The proportional text size

- **fontWeight**
  
  fontWeight: "normal" | "bold" (optional)
  
  Normal or bold text.

- **justifyItems**
  
  justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)
  
  This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

- **label**
  
  label: string (optional)

- **labelPosition**
  
  labelPosition: "stacked" | "hidden" | "inline" (optional)
  
  Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

- **name**
  
  name: string (optional)

- **padding**
  
  padding: "none" | "small" | "std" (optional)
  
  Allows specifying the component’s padding behavior. A component will inherit the padding behavior specified by its parent container components.
**type**

Type: `ControlType` (optional)

The type of the control as a string.

Inherited from `Design.type`
Hyperlink metadata type.

**Hierarchy**

ValueMetadata
  └ HyperLinkMetadata

**Index**

**Properties**

- **BoundEntity**
  - `BoundEntity` string (optional)
  
  The entity to which the control is bound.

  Inherited from `ControlMetadata.BoundEntity`

- **BoundField**
  - `BoundField` string (optional)

  Inherited from `ControlMetadata.BoundField`

- **Description**
  - `Description` string (optional)

  Description of the control.

  Inherited from `ControlMetadata.Description`
**Editable**
Editable: boolean (optional)
Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control’s edit-ability and its parent’s edit-ability is undefined.

Inherited from `ControlMetadataEditable`

**ExtType**
ExtType: `ControlType` (optional)
The extended control type. For example, a control of type Input might have an extended type of Barcode.

Inherited from `ControlMetadataExtType`

**HelpText**
HelpText: string (optional)
The keyboard shortcut for a command. For example, "(Shift+F5)"

Inherited from `ControlMetadataHelpText`

**Hidden**
Hidden: boolean (optional)
Boolean indicating if the control is hidden or not.

Inherited from `ControlMetadataHidden`

**Id**
Id: string (optional)
Identification string for a control.

Inherited from `ControlMetadataId`

**Label**
Label: string (optional)
Label for a control. For example, a control representing a person's first name might have a label "First Name".

Inherited from `ControlMetadataLabel`

**Mandatory**
Mandatory: boolean (optional)
If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline.

Inherited from `InputControlMetadataMandatory`

**Name**
Name: string (optional)
### Name of a control.

| Inherited from | [ControlMetadata.Name](#) |

---

### NumSequence

**NumSequence**: `NumberSequenceConfig` (optional)

Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Example:

```javascript
// hide number sequence reference page from users
metadataService.hideNavigation('numSeqReferencePage');

// parameters to be passed to 'numSequence' flag in configureControl
var configParam = {
  referencePageName: 'numSeqReferencePage',
  dataType: 'HcmPersonnelNumberId'
};

// setup 'PersonnelNumber' control as number sequence in the task 'add-worker'
metadataService.configureControl('add-worker', 'PersonnelNumber', { numSequence: configParam });
```

| Inherited from | [InputControlMetadata.NumSequence](#) |

---

### Order

**Order**: number (optional)

Number indicating the order in which a control will appear on a page.

| Inherited from | [ControlMetadata.Order](#) |

---

### Type

**Type**: `ControlType` (optional)

String indicating the control type.

| Inherited from | [ControlMetadata.Type](#) |
Image control interface for representing images in the mobile app. Images can be of any of the following types: DataUri, Base64, URL, AOTResource, or Symbol.

**Hierarchy**

Control
  └— Image

**Index**

**Properties**
- container
- generic
- getDataSource
- hidden
- imageSource
- imageView
- placeholderClass
- symbol

**Methods**
- applyDesign
- dataContext
- getDesign
- isEditable
- metadata
- parent
- root

**Properties**

**container**

container: boolean (optional)

True if the control is a container.

Inherited from Control.container

**generic**

generic: boolean (optional)

Inherited from Control.generic

**getDataSource**

getAddress: function(): any
hidden
hidden: boolean
True if the control is hidden.

imageSource
imageSource: string
Defines the imageSource.

imageView
imageView: string
Dictates the style of the image.

placeholderClass
placeholderClass: string

symbol
symbol: string
Defines the symbol if the image is of type symbol.

Methods

applyDesign
applyDesign(design: ImageDesign): void
Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

Overrides Control.applyDesign

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>design</td>
<td>ImageDesign</td>
<td>object containing design properties as keys</td>
</tr>
</tbody>
</table>

Returns void
dataContext
dataContext(): any

Inherited from Control.dataContext

Returns any
getDesign
getDesign(): Design

Returns the design object of this control.
isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control’s editability and its parent’s editability is undefined.

metadata(): ImageMetadata

Returns the metadata object of this control.

parent(): Control | Page

Returns the parent (control or page) of this control.

root(): Page

Returns the root form instance (page) of this control.
Image design object type.

**Hierarchy**

- Design
  - ImageDesign

**Index**

**Properties**

- alignItems
- alignSelf
- bindings
- border
- color
- flexFlow
- flexSize
- fontSize
- fontWeight
- height
- imageUrl
- justifyItems
- label
- labelPosition
- name
- padding
- type
- width

**Properties**

_**alignItems**_

**alignItems** property is an alias for the CSS property "align-items". Please refer to this web page for documentation on the "align-items" property.

- Inherited from Design.alignItems

_**alignSelf**_

**alignSelf** property is an alias for the CSS property "align-self". Please refer to this web page for documentation on the "align-self" property.

- Inherited from Design.alignSelf

_**bindings**_


**bindings: any (optional)**

Inherited from Design.bindings

**border**

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

Inherited from Design.border

**color**

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.

Consider setting the background color at the same time as necessary when setting this attribute.

Note: if color is set to "theme", the theme color of the app will be used.

The following colors are available:

- blue: #0078D7
- pomegranate: #911844
- raspberry: #8D398F
- darkOrange: #D24726
- green: #369F47
- blueberry: #17234E
- grape: #432158
- lightBlue: #5DB2FF
- lightGreen: #82BA00
- pink: #DC4FAD
- teal: #008299
- mediumDarkBlue: #004B8B
- cordovan: #570000
- darkCordovan: #380000
- black: #000000
- lightGray: #e8e8e8
- light: #fff
- dark: #333333

Inherited from Design.color

**flexFlow**

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

Inherited from Design.flexFlow
**flexSize**

flexSize: string (optional)

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

Inherited from Design.flexSize

**fontSize**

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

Inherited from Design.fontSize

**fontWeight**

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

Inherited from Design.fontWeight

**height**

height: string (optional)

The relative vertical size of the image. Sizes are about equivalent to CSS em sizes.

**imageStyle**

imageStyle: ImageStyleType (optional)

The style of the image.

**justifyItems**

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

Inherited from Design.justifyItems

**label**

label: string (optional)

Inherited from Design.label

**labelPosition**

labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

Inherited from Design.labelPosition

**name**
name: string (optional)

Inherited from Design.name

**padding**

padding: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

Inherited from Design.padding

**type**

type: ControlType (optional)

The type of the control as a string.

Inherited from Design.type

**width**

width: string (optional)

The relative horizontal size of the image. Sizes are about equivalent to CSS em sizes.
Image metadata type.

**Hierarchy**

ControlMetadata

  └ ImageMetadata

**Index**

**Properties**

- **BaseUrl**
  
  BaseUrl: string (optional)

  Base URL for AOTResource type image.

- **BoundEntity**
  
  BoundEntity: string (optional)

  The entity to which the control is bound.

  Inherited from ControlMetadata.BoundEntity

- **BoundField**
  
  BoundField: string (optional)

  Inherited from ControlMetadata.BoundField
**Description**
Description: string (optional)
Description of the control.

Inherited from ControlMetadata.Description

**Editable**
Editable: boolean (optional)
Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

Inherited from ControlMetadata.Edible

**ExtType**
ExtType: ControlType (optional)
The extended control type. For example, a control of type Input might have an extended type of Barcode.

Inherited from ControlMetadata.ExtType

**Height**
Height: number (optional)
The relative vertical size of the image. Sizes are about equivalent to CSS em sizes.

**HelpText**
HelpText: string (optional)
The keyboard shortcut for a command. For example, "(Shift+F5)"

Inherited from ControlMetadata.HelpText

**Hidden**
Hidden: boolean (optional)
Boolean indicating if the control is hidden or not.

Inherited from ControlMetadata.Hidden

**Id**
Id: string (optional)
Identification string for a control.

Inherited from ControlMetadata.Id

**ImageStyle**
ImageStyle: ImageStyleType (optional)
The style of the image.

**Label**
Label: string (optional)
Label for a control. For example, a control representing a person's first name might have a label "First Name".

| Inherited from ControlMetadata.Label |

Name
Name: string (optional)
Name of a control.

| Inherited from ControlMetadata.Name |

Order
Order: number (optional)
Number indicating the order in which a control will appear on a page.

| Inherited from ControlMetadata.Order |

Type
Type: ControlType (optional)
String indicating the control type.

| Inherited from ControlMetadata.Type |

Width
Width: number (optional)
The relative horizontal size of the image. Sizes are about equivalent to CSS em sizes.
Input control interface with methods and attributes for all input controls. Input controls are typically used on task pages for collecting user input, for example, for a new control.

**Hierarchy**

- Control
  - InputControl
    - Field
    - Lookup
    - MultiLookup
    - Value

**Index**

**Properties**
- container
- generic
- getDataSource
- hidden

**Methods**
- applyDesign
- dataContext
- getDesign
- isEditable
- metadata
- parent
- root

**Events**
- onDataChange

**Properties**

**container**
container: boolean (optional)
True if the control is a container.

| Inherited from | Control.container |

**generic**
generic: boolean (optional)

| Inherited from | Control.generic |
**getDataSource**

getDataSource: function(): any

Inherited from Control.getDataSource

**hidden**

hidden: boolean

True if the control is hidden.

Inherited from Control.hidden

## Methods

### applyDesign

applyDesign(design: Design): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

Inherited from Control.applyDesign

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>design</td>
<td>Design</td>
<td>object containing design properties as keys</td>
</tr>
</tbody>
</table>

Returns void

### dataContext

dataContext(): any

Inherited from Control.dataContext

Returns any

### getDesign

getDesign(): Design

Returns the design object of this control.

Inherited from Control.getDesign

Returns Design

### isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

Inherited from Control.isEditable
Returns boolean

metadata

metadata(): InputControlMetadata

Returns the metadata object of this control.

| Overrides Control.metadata |

Returns InputControlMetadata

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

| Inherited from Control.parent |

Returns Control | Page

root

root(): Page

Returns the root form instance (page) of this control.

| Inherited from Control.root |

Returns Page

Events

onDataChanged

onDataChanged: EventHook <null>

An event that is triggered when the input control's data changes.
Input control design.

**Hierarchy**

```
Hierarchy

Design
└─ InputControlDesign
  └─ FieldDesign
  └─ LookupDesign
  └─ MultiLookupDesign
  └─ ValueDesign
```

**Index**

**Properties**

- alignItems
- alignSelf
- bindings
- border
- color
- flexGrow
- flexSize
- fontSize
- fontWeight
- justifyItems
- label
- labelPosition
- name
- padding
- type

**Properties**

**alignItems**

alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to [this web page](https://example.com) for documentation on the "align-items" property.

- Inherited from Design.alignItems

**alignSelf**

alignSelf: string (optional)

- Inherited from Design.alignSelf
**bindings**

bindings: any (optional)

| Inherited from Design.bindings |

**border**

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

| Inherited from Design.border |

**color**

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.

Consider setting the background color at the same time as necessary when setting this attribute.

Note: if color is set to "theme", the theme color of the app will be used.

The following colors are available:

- blue: 🔊 #0078D7;
- pomegranate: 🔊 #911844;
- raspberry: 🔊 #8D398F;
- darkOrange: 🔊 #D24726;
- green: 🔊 #369F47;
- blueberry: 🔊 #17234E;
- grape: 🔊 #432158;
- lightBlue: 🔊 #5DB2FF;
- lightGreen: 🔊 #82BA00;
- pink: 🔊 #DC4FAD;
- teal: 🔊 #008299;
- mediumDarkBlue: 🔊 #004B8B;
- cordovan: 🔊 #570000;
- darkCordovan: 🔊 #380000;
- black: 🔊 #000000;
- lightGray: 🔊 #e8e8e8;
- light: 🔊 #fff;
- dark: 🔊 #333333;

| Inherited from Design.color |

**flexFlow**

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.
Inherited from Design.flexFlow

**flexSize**

`flexSize: string (optional)`

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

Inherited from Design.flexSize

**fontSize**

`fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)`

The proportional text size

Inherited from Design.fontSize

**fontWeight**

`fontWeight: "normal" | "bold" (optional)`

Normal or bold text.

Inherited from Design.fontWeight

**justifyItems**

`justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)`

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

Inherited from Design.justifyItems

**label**

`label: string (optional)`

Inherited from Design.label

**labelPosition**

`labelPosition: "stacked" | "hidden" | "inline" (optional)`

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

Inherited from Design.labelPosition

**name**

`name: string (optional)`

Inherited from Design.name

**padding**

`padding: "none" | "small" | "std" (optional)"
Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

| Inherited from Design.padding |

**type**

Type: `ControlType` (optional)

The type of the control as a string.

| Inherited from Design.type |
Metadata for input controls.

**Hierarchy**

ControlMetadata  
└── InputControlMetadata  
    └── FieldMetadata  
    └── LookupMetadata  
    └── MultiLookupMetadata  
    └── ValueMetadata

**Index**

**Properties**

- **BoundEntity**
- **BoundField**
- **Description**
- **Editable**
- **ExtType**
- **HelpText**
- **Hidden**
- **Id**
- **Label**
- **Mandatory**
- **Name**
- **NumSequence**
- **Order**
- **Type**

**Properties**

**BoundEntity**

BoundEntity: string (optional)

The entity to which the control is bound.

Inherited from ControlMetadata.BoundEntity

**BoundField**

BoundField: string (optional)

Inherited from ControlMetadata.BoundField

**Description**

Description: string (optional)
Description of the control.

| Inherited from ControlMetadata.Description |

**Editable**

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control’s edit-ability and its parent’s edit-ability is undefined.

| Inherited from ControlMetadata.Editable |

**ExtType**

ExtType: ControlType (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

| Inherited from ControlMetadata.ExtType |

**HelpText**

HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

| Inherited from ControlMetadata.HelpText |

**Hidden**

Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

| Inherited from ControlMetadata.Hidden |

**Id**

Id: string (optional)

Identification string for a control.

| Inherited from ControlMetadata.Id |

**Label**

Label: string (optional)

Label for a control. For example, a control representing a person’s first name might have a label "First Name".

| Inherited from ControlMetadata.Label |

**Mandatory**

Mandatory: boolean (optional)

If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline.
Name
Name: string (optional)

Name of a control.

   Inherited from ControlMetadata.Name

NumSequence
NumSequence: NumberSequenceConfig (optional)

Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Example:

```javascript
// hide number sequence reference page from users
metadataService.hideNavigation('numSeqReferencePage');

// parameters to be passed to 'numSequence' flag in configureControl
var configParam = {
    referencePageName: 'numSeqReferencePage',
    dataType: 'HcmPersonnelNumberId'
};

// setup 'PersonnelNumber' control as number sequence in the task 'add-worker'
metadataService.configureControl('add-worker', 'PersonnelNumber', { numSequence: configParam });
```

Order
Order: number (optional)

Number indicating the order in which a control will appear on a page.

   Inherited from ControlMetadata.Order

Type
Type: ControlType (optional)

String indicating the control type.

   Inherited from ControlMetadata.Type
List control type. A list is a control that contains any numbers of rows. Each row follows a template for the layout of any number of controls. Lists come in two styles: simple and card.

**Hierarchy**

ContainerControl
  └ List

**Index**

**Properties**

- $accessibility
- DefaultSearchColumn
- container
- emptyListMessage
- enableMultiSelect
- generic
- getDataSource
- hidden
- hideEmptyListMessage
- imageFields
- performingRemoteSearch
- searchQuery

**Methods**

- allowsNavigation
- applyDesign
- applySearch
- canPerformRemoteSearch
- clearSearch
- dataContext
- getColumnLabel
- getControl
- getControlById
- getControlMetadata
- getControlMetadataById
- getData
- getDesign
- getListData
- getRenderedRows
- getRowNavigation
- getRowSelectionCount
- getRowSelections
- `getRowTracking`
- `getSearchColumn`
- `getSearchColumnLabel`
- `getSearchableColumns`
- `hideSearchBar`
- `isEditable`
- `loadMetaData`
- `loadMore`
- `metadata`
- `parent`
- `performRemoteSearch`
- `root`
- `selectSearchColumn`
- `setRowSections`

**Events**
- `onRowCreate`
- `onRowSelect`

**Properties**

$accessibility
$accessibility: any

**DefaultSearchColumn**
DefaultSearchColumn: string

**container**
container: boolean
True if the control is a container.

- Inherited from `ContainerControl.container`
- Overrides `Control.container`

**emptyListMessage**
emptyListMessage: string
Settable property to override default empty list message.

**enableMultiSelect**
enableMultiSelect: boolean

**generic**
generic: boolean (optional)

- Inherited from `Control.generic`

**getDataSource**
getDataSource: function(): any

- Inherited from `Control.getDataSource`
**hidden**

hidden: boolean

True if the control is hidden.

Inherited from Control.hidden

**hideEmptyListMessage**

hideEmptyListMessage: boolean

If true, no message is shown if the list is empty. To set this property, update the corresponding metadata property via configureControl.

**imageFields**

imageFields: any []

**performingRemoteSearch**

performingRemoteSearch: boolean

**searchQuery**

searchQuery: [value: string]: any

### Methods

**allowsNavigation**

allowsNavigation(): boolean

**applyDesign**

applyDesign(IDesign: ListDesign): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

Overrides Control.applyDesign

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDesign</td>
<td>ListDesign</td>
<td>object containing design properties as keys</td>
</tr>
</tbody>
</table>

**applySearch**

applySearch(): void

**canPerformRemoteSearch**

canPerformRemoteSearch(): boolean

**clearSearch**

clearSearch(): void
**Returns void**

dataContext
dataContext(): any

Inherited from `Control.dataContext`

**Returns any**

getColumnLabel
getColumnLabel(id: string): string

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

**Returns string**

getControl
getControl(controlName: string): `Control`

Given the name of a control, returns the control instance.

Inherited from `ContainerControl.getControl`

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlName</td>
<td>string</td>
<td>control name</td>
</tr>
</tbody>
</table>

**Returns `Control`**

getControlById
getControlById(id: string): `Control`

Given the ID of a control, returns the control instance.

Inherited from `ContainerControl.getControlById`

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>control ID</td>
</tr>
</tbody>
</table>

**Returns `Control`**

getControlMetadata
getControlMetadata(controlName: string): `Control`

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlName</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>
getControlMetadataById

getControlMetadataById(id: string): Control

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

Returns Control

data

data(): any []

Returns any []

design

design(): Design

Returns the design object of this control.

Inherited from Control.getDesign

Returns Design

getListData

getListData(): any

Returns any

getRenderedRows

getRenderedRows(): Row []

Returns Row []

getRowNavigation

getRowNavigation(row: Row): Promise <any> | any

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>row</td>
<td>Row</td>
<td></td>
</tr>
</tbody>
</table>

Returns Promise <any> | any

getRowSelectionCount

getRowSelectionCount(): number

Returns number

getRowSelections

getRowSelections(): string []

Returns string []

getRowTracking

getRowTracking(row: any, index: string): string

Parameters
<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>row</td>
<td>any</td>
<td>Returns string</td>
</tr>
<tr>
<td>index</td>
<td>string</td>
<td>Returns string</td>
</tr>
</tbody>
</table>

**getSearchColumn**

getSearchColumn(): string

Returns string

**getSearchColumnLabel**

getSearchColumnLabel(): string

Returns string

**getSearchableColumns**

getSearchableColumns(): any [ ]

Returns any [ ]

**hideSearchBar**

hideSearchBar(): boolean

Returns boolean

**isEditable**

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

Inherited from Control.isEditable

**loadMetaData**

loadMetaData(): void

Returns void

**loadMore**

loadMore(): void

Returns void

**metadata**

metadata(): ListMetadata

Returns the metadata object of this control.

Overrides ContainerControl.metadata

Returns ListMetadata

**parent**
parent(): Control | Page

Returns the parent (control or page) of this control.

Inherited from Control.parent

performRemoteSearch(): void

Returns Control | Page

root(): Page

Returns the root form instance (page) of this control.

Inherited from Control.root

selectSearchColumn(column: string): void

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

Returns void

setRowSections(selections: string [ ]): void

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>selections</td>
<td>string [ ]</td>
<td></td>
</tr>
</tbody>
</table>

Returns void

Events

onRowCreate: EventHook <Row>

onRowSelect: EventHook <Row>
List design object type.

### Hierarchy

ContainerControlDesign  
  └ ListDesign

### Index

#### Properties

- `alignItems`
- `alignSelf`
- `allowScroll`
- `background`
- `bindings`
- `border`
- `color`
- `design`
- `flexFlow`
- `flexSize`
- `fontSize`
- `fontWeight`
- `hideArrow`
- `hideSearchBar`
- `itemBorder`
- `items`
- `justifyItems`
- `label`
- `labelPosition`
- `name`
- `padding`
- `type`

#### Properties

**alignItems**  
`alignItems: string (optional)`

This property is an alias for the CSS property "align-items". Please refer to [this web page](#) for documentation on the "align-items" property.

Inherited from Design.alignItems
alignSelf: string (optional)

| Inherited from Design.alignSelf |

allowScroll

allowScroll: string (optional)

True if the container will allow scrolling when its items do not fit into the container's available space. If a container has an item which may scroll, then set this property to false to prevent nested scrolling areas.

| Inherited from ContainerControlDesign.allowScroll |

background

background: string (optional)

The background color of the container. Consider modifying the color attribute in the same container so that fonts overlaying the background color will appear appropriately. Note: if background is set to "theme", the theme color of the app will be used. The following colors are available:

- **blue**: #0078D7
- **pomegranate**: #911844
- **raspberry**: #8D398F
- **darkOrange**: #D24726
- **green**: #369F47
- **blueberry**: #17234E
- **grape**: #432158
- **lightBlue**: #5DB2FF
- **lightGreen**: #82BA00
- **pink**: #DC4FAD
- **teal**: #008299
- **mediumDarkBlue**: #004B8B
- **cordovan**: #570000
- **darkCordovan**: #380000
- **black**: #000000
- **lightGray**: #e8e8e8
- **light**: #fff
- **dark**: #333333

| Inherited from ContainerControlDesign.background |

bindings

bindings: any (optional)

| Inherited from Design.bindings |

border

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)
The border behavior of a control. This property will not be inherited by the children.

| Inherited from Design.border |

**color**
color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container. Consider setting the background color at the same time as necessary when setting this attribute. Note: if color is set to "theme", the theme color of the app will be used.

The following colors are available:

- **blue**: #0078D7
- **pomegranate**: #911844
- **raspberry**: #B03060
- **darkOrange**: #D24726
- **green**: #369F47
- **blueberry**: #17238E
- **grape**: #432158
- **lightBlue**: #6DB2FF
- **lightGreen**: #82BA00
- **pink**: #DC4FAD
- **teal**: #008299
- **mediumDarkBlue**: #004B8B
- **cordovan**: #570000
- **darkCordovan**: #380000
- **black**: #000000
- **lightGray**: #e8e8e8
- **light**: #fff
- **dark**: #333333

| Inherited from Design.color |

**design**
design: GroupDesign (optional)

The design object that will be applied to each row.

**flexFlow**
flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

| Inherited from Design.flexFlow |

**flexSize**
flexSize: string (optional)

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

| Inherited from Design.flexSize |

**fontSize**

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

| Inherited from Design.fontSize |

**fontWeight**

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

| Inherited from Design.fontWeight |

**hideArrow**

hideArrow: boolean (optional)

Allows an arrow ( > ) on a default styled navigation control to be hidden. Note that if the list has DetailsPageld, navigationHandler, or OnNavigate in the metadata then by default the arrows are present in each row of the list to show that the rows are clickable.

This property can only be added through the design object.

**hideSearchBar**

hideSearchBar: boolean (optional)

If true, the search bar will be hidden.

**itemBorder**

itemBorder: "solid" | "none" (optional)

If true, a border will appear around each row in the list. This property is equivalent to applying the border property individually to all items in the container.

| Inherited from ContainerControlDesign.itemBorder |

**items**

items: string | Design [ ] (optional)

An array containing the components to place inside of the container.

| Inherited from ContainerControlDesign.items |

**justifyItems**

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation...
on the "justify-content" property.

- label
  label: string (optional)
  Inherited from Design.label

- labelPosition
  labelPosition: "stacked" | "hidden" | "inline" (optional)
  Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.
  Inherited from Design.labelPosition

- name
  name: string (optional)
  Inherited from Design.name

- padding
  padding: "none" | "small" | "std" (optional)
  Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.
  Inherited from Design.padding

- type
  type: ControlType (optional)
  The type of the control as a string.
  Inherited from Design.type
Metadata for list control.

Hierarchy

ContainerControlMetadata
  └ ListMetadata

Index

Properties

- BoundEntity
- BoundField
- Children
- Description
- DetailsPageAppId
- DetailsPageId
- Editable
- EmptyListMessage
- ExtType
- HelpText
- Hidden
- HideEmptyListMessage
- HideSearchBar
- Id
- InfiniteScroll
- InfiniteScrollPageSize
- Label
- ListStyle
- MultiSelect
- Name
- NonEntityProjection
- Order
- Type

Methods

- navigationHandler

Events

- OnNavigate
- OnRowSelect

Properties

BoundEntity
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BoundEntity</strong></td>
<td>The entity to which the control is bound.</td>
</tr>
<tr>
<td><strong>BoundField</strong></td>
<td><strong>BoundField</strong>: string (optional)</td>
</tr>
<tr>
<td><strong>Children</strong></td>
<td><strong>Children</strong>: ControlMetadata [] (optional)</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td><strong>Description</strong>: string (optional)</td>
</tr>
<tr>
<td><strong>DetailsPageAppId</strong></td>
<td><strong>DetailsPageAppId</strong>: string (optional)</td>
</tr>
<tr>
<td><strong>DetailsPageId</strong></td>
<td><strong>DetailsPageId</strong>: string (optional)</td>
</tr>
<tr>
<td><strong>Editable</strong></td>
<td>Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control’s edit-ability and its parent's edit-ability is undefined.</td>
</tr>
<tr>
<td><strong>EmptyListMessage</strong></td>
<td><strong>EmptyListMessage</strong>: string (optional)</td>
</tr>
<tr>
<td><strong>ExtType</strong></td>
<td><strong>ExtType</strong>: ControlType (optional)</td>
</tr>
<tr>
<td><strong>HelpText</strong></td>
<td><strong>HelpText</strong>: string (optional)</td>
</tr>
</tbody>
</table>
The keyboard shortcut for a command. For example, "(Shift+F5)"

| Inherited from ControlMetadata.HelpText |

**Hidden**

Hidden boolean (optional)

Boolean indicating if the control is hidden or not.

| Inherited from ControlMetadata.Hidden |

**HideEmptyListMessage**

HideEmptyListMessage boolean (optional)

If true, the empty list message will be hidden.

**HideSearchBar**

HideSearchBar boolean (optional)

If true, the search bar will be hidden.

**Id**

Id string (optional)

Identification string for a control.

| Inherited from ControlMetadata.Id |

**InfiniteScroll**

InfiniteScroll boolean (optional)

If set to true then the list will allow infinite scroll.

**InfiniteScrollPageSize**

InfiniteScrollPageSize number (optional)

Number of rows to load initially and the number of rows to load after the user reaches the end of the currently displayed rows.

**Label**

Label string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

| Inherited from ControlMetadata.Label |

**ListStyle**

ListStyle string (optional)

Dictates the list template type. Options:

- "Simple": simple style
- "Card": card style

**MultiSelect**

MultiSelect boolean (optional)

If true, then the list will be a multi-select list.

**Name**

Name: string (optional)

Name of a control.

<table>
<thead>
<tr>
<th>Inherited from</th>
<th>ControlMetadata.Name</th>
</tr>
</thead>
</table>

**NonEntityProjection**

NonEntityProjection: boolean (optional)

**Order**

Order: number (optional)

Number indicating the order in which a control will appear on a page.

<table>
<thead>
<tr>
<th>Inherited from</th>
<th>ControlMetadata.Order</th>
</tr>
</thead>
</table>

**Type**

Type: ControlType (optional)

String indicating the control type.

<table>
<thead>
<tr>
<th>Inherited from</th>
<th>ControlMetadata.Type</th>
</tr>
</thead>
</table>

**Methods**

**navigationHandler**

Optional

navigationHandler(row: Row): Promise <any> | NavigationArgs

A function that determines the navigation for a given row.

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>row</td>
<td>Row</td>
<td>row to get navigation handler for.</td>
</tr>
</tbody>
</table>

| Returns | Promise <any> | NavigationArgs |

**Events**

**OnNavigate**


An event that is triggered when a pagelink control is selected.

**OnRowSelect**

OnRowSelect: function(row: Row): void (optional)

An event that is triggered when a row is selected.
Lookup control type. A lookup is an input control that is used to select an input from a list of options. For example, a lookup could be used to lookup a customer when linking a customer to a new sales order.

**Hierarchy**

```
InputControl
  └ Lookup
```

**Index**

**Properties**
- `container`
- `generic`
- `dataSource`
- `hidden`

**Methods**
- `applyDesign`
- `dataSource`
- `getDesign`
- `getDisplayValue`
- `getLookupPage`
- `getValue`
- `isEditable`
- `metadata`
- `parent`
- `root`
- `setEntityRef`

**Events**
- `onDataChanged`

**Properties**

**container**

container: boolean (optional)

True if the control is a container.

Inherited from `Control.container`

**generic**

generic: boolean (optional)

Inherited from `Control.generic`
**getDataSource**
getDataSource: function(): any

Inherited from Control.getDataSource

**hidden**
hidden: boolean

True if the control is hidden.

Inherited from Control.hidden

**Methods**

**applyDesign**
applyDesign(IDesign: LookupDesign): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

Overrides Control.applyDesign

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDesign</td>
<td>LookupDesign</td>
<td>object containing design properties as keys</td>
</tr>
</tbody>
</table>

Returns void

**dataContext**

dataContext(): any

Inherited from Control.dataContext

Returns any

**getDesign**

getDesign(): Design

Returns the design object of this control.

Inherited from Control.getDesign

Returns Design

**getDisplayValue**

getDisplayValue(): string

Returns string

**getLookupPage**

getLookupPage(): Page

Returns Page

**getValue**
getValue(): string | number

Returns string | number

isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

Inherited from Control.isEditable

Returns boolean

metadata

metadata(): LookupMetadata

Returns the metadata object of this control.

Overrides InputControl.metadata

Returns LookupMetadata

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

Inherited from Control.parent

Returns Control | Page

root

root(): Page

Returns the root form instance (page) of this control.

Inherited from Control.root

Returns Page

setEntityRef

setEntityRef(newValue: string | number): Promise <any>

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>newValue</td>
<td>string</td>
<td>number</td>
</tr>
</tbody>
</table>

Returns Promise <any>

Events

onDataChanged
onDataChanged: EventHook <null>

An event that is triggered when the input control's data changes.

Inherited from InputControl.onDataChanged
Lookup design object type.

Hierarchy
InputControlDesign
  └ LookupDesign

Index

Properties
- alignItems
- alignSelf
- bindings
- border
- color
- flexGrow
- flexSize
- fontWeight
- justifyItems
- label
- labelPosition
- name
- padding
- type

Properties

alignItems
alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to this web page for documentation on the "align-items" property.

Inherited from Design.alignItems

alignSelf
alignSelf: string (optional)

Inherited from Design.alignSelf

bindings
bindings: any (optional)

Inherited from Design.bindings
**border**

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

Inherited from Design.border

**color**

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.

Consider setting the background color at the same time as necessary when setting this attribute.

Note: if color is set to "theme", the theme color of the app will be used.

The following colors are available:

- **blue**: #0078D7;
- **pomegranate**: #911844;
- **raspberry**: #8D398F;
- **darkOrange**: #D24726;
- **green**: #369F47;
- **blueberry**: #1723E;
- **grape**: #432158;
- **lightBlue**: #5DB2FF;
- **lightGreen**: #82BA00;
- **pink**: #DC4FAD;
- **teal**: #008299;
- **mediumDarkBlue**: #04B8B;
- **cordovan**: #570000;
- **darkCordovan**: #380000;
- **black**: #000000;
- **lightGray**: #8e8e8e;
- **light**: #fff;
- **dark**: #333333;

Inherited from Design.color

**flexFlow**

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

Inherited from Design.flexFlow

**flexSize**

flexSize: string (optional)
One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to [this web page](#) for documentation on the "flex" property.

<table>
<thead>
<tr>
<th>Inherited from</th>
<th>Design.flexSize</th>
</tr>
</thead>
</table>

**fontSize**

`font-size: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)`

The proportional text size

<table>
<thead>
<tr>
<th>Inherited from</th>
<th>Design.fontSize</th>
</tr>
</thead>
</table>

**fontWeight**

`font-weight: "normal" | "bold" (optional)`

Normal or bold text.

<table>
<thead>
<tr>
<th>Inherited from</th>
<th>Design.fontWeight</th>
</tr>
</thead>
</table>

**justifyItems**

`justify-items: "flex-start" | "flex-end" | "center" | "space-between" (optional)`

This property is an alias for the CSS property "justify-content". Please refer to [this web page](#) for documentation on the "justify-content" property.

<table>
<thead>
<tr>
<th>Inherited from</th>
<th>Design.justifyItems</th>
</tr>
</thead>
</table>

**label**

`label: string (optional)`

<table>
<thead>
<tr>
<th>Inherited from</th>
<th>Design.label</th>
</tr>
</thead>
</table>

**labelPosition**

`label-position: "stacked" | "hidden" | "inline" (optional)`

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

<table>
<thead>
<tr>
<th>Inherited from</th>
<th>Design.labelPosition</th>
</tr>
</thead>
</table>

**name**

`name: string (optional)`

<table>
<thead>
<tr>
<th>Inherited from</th>
<th>Design.name</th>
</tr>
</thead>
</table>

**padding**

`padding: "none" | "small" | "std" (optional)`

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

<table>
<thead>
<tr>
<th>Inherited from</th>
<th>Design.padding</th>
</tr>
</thead>
</table>
**type**

type: `ControlType` (optional)

The type of the control as a string.

| Inherited from Design.type |
Lookup metadata type.

Hierarchy
InputControlMetadata
  └ LookupMetadata

Index

Properties
• BoundEntity
• BoundField
• Description
• DisplayField
• DisplayKey
• Editable
• ExtType
• FilterContext
• HelpText
• Hidden
• Id
• Label
• LookupEntity
• LookupPage
• LookupPageld
• Mandatory
• MultiSelect
• Name
• NumSequence
• Order
• ReferenceAppId
• ShowLookupPage
• Type
• ValueField
• ValueKey

Events
• OnOptionSelected
• OnValueChanged

Properties

BoundEntity
BoundEntity: string (optional)
The entity to which the control is bound.

Inherited from `ControlMetadata.BoundEntity`

**BoundField**

BoundField: string (optional)

Inherited from `ControlMetadata.BoundField`

**Description**

Description: string (optional)

Description of the control.

Inherited from `ControlMetadata.Description`

**DisplayField**

DisplayField: string (optional)

The name of a control on the page, whose value should be displayed to the user. Usually, this value is user-friendly/user-readable text.

**DisplayKey**

DisplayKey: string (optional)

**Editable**

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control’s edit-ability and its parent’s edit-ability is undefined.

Inherited from `ControlMetadata.Editable`

**ExtType**

ExtType: `ControlType` (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

Inherited from `ControlMetadata.ExtType`

**FilterContext**

FilterContext: DataFilter (optional)

**HelpText**

HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

Inherited from `ControlMetadata.HelpText`

**Hidden**

Hidden: boolean (optional)
Boolean indicating if the control is hidden or not.

| Inherited from ControlMetadata.Hidden |

**Id**

Id: string (optional)

Identification string for a control.

| Inherited from ControlMetadata.Id |

**Label**

Label: string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

| Inherited from ControlMetadata.Label |

**LookupEntity**

LookupEntity: any (optional)

The entity that is being looked up in the lookup.

**LookupPage**

LookupPage: string (optional)

**LookupPageId**

LookupPageId: string (optional)

**Mandatory**

Mandatory: boolean (optional)

If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline.

| Inherited from InputControlMetadata.Mandatory |

**MultiSelect**

MultiSelect: boolean (optional)

If true, lookup will be configured as a multi-select.

**Name**

Name: string (optional)

Name of a control.

| Inherited from ControlMetadata.Name |

**NumSequence**

NumSequence: NumberSequenceConfig (optional)

Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Example:
// hide number sequence reference page from users
metadataService.hideNavigation('numSeqReferencePage');

// parameters to be passed to 'numSequence' flag in configureControl
var configParam = {
    referencePageName: 'numSeqReferencePage',
    dataType: 'HcmPersonnelNumberId'
};

// setup 'PersonnelNumber' control as number sequence in the task 'add-worker'
metadataService.configureControl('add-worker', 'PersonnelNumber', { numSequence: configParam });

Inherited from InputControlMetadata.NumSequence

Order

Order: number (optional)

Number indicating the order in which a control will appear on a page.

Inherited from ControlMetadata.Order

ReferenceAppId

ReferenceAppId: string (optional)

ShowLookupPage

ShowLookupPage: boolean (optional)

Type

Type: ControlType (optional)

String indicating the control type.

Inherited from ControlMetadata.Type

ValueField

ValueField: string (optional)

The name of a control on the page, whose value should be used when committing the data. Usually, this value is a unique key.

ValueKey

ValueKey: string (optional)

Events

OnOptionSelected

OnOptionSelected: function(lookup: any, lookupEntityData: any): void (optional)

An event that is triggered by an option being selected.

OnValueChanged

OnValueChanged: function(value: any): void (optional)

An event that is triggered by a value being changed.
Provides ability to access and configure various metadata elements under the application workspace.

**Hierarchy**
MetadataService

**Index**

**Properties**

- **version**

**Methods**

- **addControl**
- **compareVersion**
- **configureAction**
- **configureControl**
- **configureEntity**
- **configureLookup**
- **configurePage**
- **configureWorkspace**
- **findAction**
- **findControl**
- **findPage**
- **getFilterExpression**
- **getFormReference**
- **hideNavigation**

**Properties**

**version**

version: string

(Read-only) Gets the version of the platform currently running.

**Methods**

**addControl**

addControl(componentName: string, controlName: string, controlType: ControlType, parentContainerName?: string, options?: ControlMetadata): any

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>componentName</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>controlName</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>controlType</td>
<td>ControlType</td>
<td></td>
</tr>
<tr>
<td>parentContainerName?</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>options?</td>
<td>ControlMetadata</td>
<td></td>
</tr>
</tbody>
</table>

**compareVersion**

 Returns any

`compareVersion(versionToCompare: string): 1 | -1`

Compares the current platform version with a reference version.

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>versionToCompare</td>
<td>string</td>
<td>The reference version to compare with</td>
</tr>
</tbody>
</table>

**Returns 1 | -1**

1 to indicate the platform version is older than the reference version, -1 to indicate that the platform version is newer or same as the reference version

**configureAction**

`configureAction(actionName: string, options: PageMetadata): any`

Configuring an action allows specifying or overriding certain behaviors specific to actions. Example:

```
metadataService.configureAction('Edit-Reservation', { properties-to-set });
```

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>actionName</td>
<td>string</td>
<td>The action whose behavior is to be changed</td>
</tr>
<tr>
<td>options</td>
<td>PageMetadata</td>
<td>The property bag containing the properties to set on the action</td>
</tr>
</tbody>
</table>

**Returns any**

**configureControl**

`configureControl(componentName: string, controlName: string, options: ControlMetadata): any`

Configuring a control allows specifying or overriding certain behaviors specific to the control. Note that the available behaviors vary by control type. Example:

```
metadataService.configureControl('All-Customers', 'FMCustomer_RecId', { properties-to-set });
```

**Parameters**
<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>componentName</td>
<td>string</td>
<td>A page or action that contains the control</td>
</tr>
<tr>
<td>controlName</td>
<td>string</td>
<td>The control whose behavior is to be changed</td>
</tr>
<tr>
<td>options</td>
<td>ControlMetadata</td>
<td>The property bag containing the properties to set on the control</td>
</tr>
</tbody>
</table>

Returns any

**configureEntity**
configureEntity(entityName: string, options: any): any

Configuring an entity allows specifying or overriding certain behaviors specific to the entity. Example:

```javascript
metadataService.configureEntity("FMCustomer", { properties-to-set });
```

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>entityName</td>
<td>string</td>
<td>An entity name</td>
</tr>
<tr>
<td>options</td>
<td>any</td>
<td>The property bag containing the properties to set on the entity</td>
</tr>
</tbody>
</table>

Returns any

**configureLookup**
configureLookup(taskName: string, lookupControlName: string, options: LookupMetadata): any

Configures a field on an action to behave as a lookup. Requires using an existing page which contains a list control. Example:

```javascript
metadataService.configureLookup('Add-Reservation', 'FMRental_Customer', { lookupPage: 'All-Customers', valueField: 'FMCustomer_RecId', displayField: 'FMCustomer_FullName'});
```

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>taskName</td>
<td>string</td>
<td>Action name</td>
</tr>
<tr>
<td>lookupControlName</td>
<td>string</td>
<td>The control name of the field to be given lookup behavior</td>
</tr>
<tr>
<td>options</td>
<td>LookupMetadata</td>
<td>Lookup configuration object</td>
</tr>
</tbody>
</table>

Returns any

**configurePage**
configurePage(pageName: string, options: PageMetadata): any

Configuring a Page allows specifying or overriding certain behaviors specific to the Page. Example:
metadataService.configurePage('Reservation-details', { properties-to-set });

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>pageName</td>
<td>string</td>
<td>The page that contains the control</td>
</tr>
<tr>
<td>options</td>
<td>PageMetadata</td>
<td>The property bag containing the properties to set on the page</td>
</tr>
</tbody>
</table>

**Returns any**

**configureWorkspace**

configureWorkspace(options: PageMetadata): any

Configuring a workspace allows specifying or overriding certain behaviors specific to the workspace. Example:

```javascript
metadataService.configureWorkspace({ properties-to-set });
```

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>options</td>
<td>PageMetadata</td>
<td>The property bag containing the properties to set on the workspace</td>
</tr>
</tbody>
</table>

**Returns any**

**findAction**

findAction(actionName: string): PageMetadata

Gets a copy of the current metadata instance of a specified Action, for the purpose of inspecting the metadata (not to be used for changing the metadata). Note: Since metadata can be changed at any time by business logic, you must be mindful of when you use this API to get a copy as it will reflect the state of the metadata at the time the call is made.

Example:

```javascript
var newCustomerTaskMetadata = metadataService.findTask("New-customer");
```

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>actionName</td>
<td>string</td>
<td>An action name</td>
</tr>
</tbody>
</table>

**Returns** PageMetadata

**findControl**

findControl(componentMetadata: any, controlName: string): ControlMetadata

Gets a copy of the current metadata instance of a specified control, for the purpose of inspecting the metadata (not to be used for changing the metadata). Note: Since metadata can be changed at any time by business logic, you must be mindful of when you use this API to get a copy as it will reflect the state of the metadata at the time the call is made.
Example:

```javascript
var firstNameControl = metadataService.findControl(newCustomerTaskMetadata, 'FMCustomer_FirstName');
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>componentMetadata</td>
<td>any</td>
<td>A metadata instance of the page or action</td>
</tr>
<tr>
<td>controlName</td>
<td>string</td>
<td>A control name</td>
</tr>
</tbody>
</table>

**Returns** [ControlMetadata](#)

### findPage

**findPage**(pageName: string): [PageMetadata](#)

Gets a copy of the current metadata instance of a specified page, for the purpose of inspecting the metadata (not to be used for changing the metadata). Note: Since metadata can be changed at any time by business logic, you must be mindful of when you use this API to get a copy as it will reflect the state of the metadata at the time the call is made.

Example:

```javascript
var reservationDetailsMetadata = metadataService.findPage("Reservation-details");
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>pageName</td>
<td>string</td>
<td>A page name</td>
</tr>
</tbody>
</table>

**Returns** [PageMetadata](#)

### getFilterExpression

**getFilterExpression**(pageName: string, listControlName: string, controlName: string, operator: [ExpressionOperator](#), value: string): DataFilter

Create a DataFilter object for a list control based on the provided options. Example:

```javascript
var filter = metadataService.getFilterExpression(
    pageNames.AllCustomers, controlNames.Customerlist, controlNames.CustomerFullName, "Is", firstCustomerName),
```

<table>
<thead>
<tr>
<th>Parameters</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>pageName</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>listControlName</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>controlName</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>operator</td>
<td><a href="#">ExpressionOperator</a></td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

**Returns** `DataFilter`

**getFormReference**

getFormReference(componentName: string, filterContext: DataFilter, excludeContext: boolean, filterLocalOnly?: boolean): `NavigationArgs`

Create an `INavigationArgs` object for a specific page/action to be used with a navigation control.

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>componentName</td>
<td>string</td>
<td>Name of the action/page</td>
</tr>
<tr>
<td>filterContext</td>
<td>DataFilter</td>
<td></td>
</tr>
<tr>
<td>excludeContext</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filterLocalOnly?</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

**Returns** `NavigationArgs`

**hideNavigation**

hideNavigation(pageNamesToHide: string[]): any

Hides the specified page(s) from the default landing page. Example:

```javascript
metadataService.hideNavigation('Select-a-customer', 'Select-a-vehicle');
```

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>pageNamesToHide</td>
<td>string</td>
<td>Page name(s)</td>
</tr>
</tbody>
</table>

**Returns** any
Multi-Lookup control type. Multi-Lookup controls are similar to regular lookups except they allow multiple selections at once.

**Hierarchy**

```
InputControl
    └ MultiLookup
```

**Index**

**Properties**

- `container`
- `generic`
- `getDataSource`
- `getEntityRefs`
- `hidden`
- `setEntityRefs`

**Methods**

- `applyDesign`
- `dataContext`
- `getDesign`
- `getLookupPage`
- `isEditable`
- `metadata`
- `parent`
- `root`

**Events**

- `onDataChanged`

**Properties**

- `container`
  
  container: boolean (optional)

  True if the control is a container.

  Inherited from `Control.container`

- `generic`
  
  generic: boolean (optional)

  Inherited from `Control.generic`

- `getDataSource`
**getDataSource:** function(): any

Inherited from Control.getDataSource

**getEntityRefs**

getEntityRefs: function(): string[] | number[]

**hidden**

hidden: boolean

True if the control is hidden.

Inherited from Control.hidden

**setEntityRefs**

setEntityRefs: function(ids: string[] | number[]): Promise <any>

**Methods**

**applyDesign**

applyDesign(IDesign: MultiLookupDesign): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

Overrides Control.applyDesign

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDesign</td>
<td>MultiLookupDesign</td>
<td>object containing design properties as keys</td>
</tr>
</tbody>
</table>

**Returns** void

**dataContext**

dataContext(): any

Inherited from Control.dataContext

**Returns** any

**getDesign**

getDesign(): Design

Returns the design object of this control.

Inherited from Control.getDesign

**Returns** Design

**getLookupPage**

getLookupPage(): Page

**Returns** Page
isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control’s edit-ability and its parent’s edit-ability is undefined.

Inherited from Control.isEditable

metadata

metadata(): MultiLookupMetadata

Returns the metadata object of this control.

Overrides InputControl.metadata

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

Inherited from Control.parent

root

root(): Page

Returns the root form instance (page) of this control.

Inherited from Control.root

Events

onDataChanged

onDataChanged: EventHook <null>

An event that is triggered when the input control’s data changes.

Inherited from InputControl.onDataChanged
Multi-Lookup design object type.

**Hierarchy**

- InputControlDesign
- MultiLookupDesign

**Index**

**Properties**

- `alignItems`
- `alignSelf`
- `bindings`
- `border`
- `color`
- `flexFlow`
- `flexSize`
- `fontSize`
- `fontWeight`
- `justifyItems`
- `label`
- `labelPosition`
- `name`
- `padding`
- `type`

**Properties**

**alignItems**

`alignItems: string (optional)`

This property is an alias for the CSS property "align-items". Please refer to [this web page](#) for documentation on the "align-items" property.

Inherited from `Design.alignItems`

**alignSelf**

`alignSelf: string (optional)`

Inherited from `Design.alignSelf`

**bindings**

`bindings: any (optional)`

Inherited from `Design.bindings`
**border**

`border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)`

The border behavior of a control. This property will not be inherited by the children.

Inherited from Design.border

**color**

`color: string (optional)`

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.

Consider setting the background color at the same time as necessary when setting this attribute.

Note: if color is set to "theme", the theme color of the app will be used.

The following colors are available:

- **blue**: #0078D7
- **pomegranate**: #911844
- **raspberry**: #8D398F
- **darkOrange**: #D24726
- **green**: #369F47
- **blueberry**: #17234E
- **grape**: #432158
- **lightBlue**: #5DB2FF
- **lightGreen**: #82BA00
- **pink**: #DC4FAD
- **teal**: #008299
- **mediumDarkBlue**: #004B8B
- **cordovan**: #570000
- **darkCordovan**: #380000
- **black**: #000000
- **lightGray**: #e8e8e8
- **light**: #fff
- **dark**: #333333

Inherited from Design.color

**flexFlow**

`flexFlow: string (optional)`

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.

Inherited from Design.flexFlow

**flexSize**

`flexSize: string (optional)`
One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to [this web page](#) for documentation on the "flex" property.

<table>
<thead>
<tr>
<th>Inherited from Design.flexSize</th>
</tr>
</thead>
</table>

**fontSize**

`fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)`

The proportional text size

<table>
<thead>
<tr>
<th>Inherited from Design.fontSize</th>
</tr>
</thead>
</table>

**fontWeight**

`fontWeight: "normal" | "bold" (optional)`

Normal or bold text.

<table>
<thead>
<tr>
<th>Inherited from Design.fontWeight</th>
</tr>
</thead>
</table>

**justifyItems**

`justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)`

This property is an alias for the CSS property "justify-content". Please refer to [this web page](#) for documentation on the "justify-content" property.

<table>
<thead>
<tr>
<th>Inherited from Design.justifyItems</th>
</tr>
</thead>
</table>

**label**

`label: string (optional)`

<table>
<thead>
<tr>
<th>Inherited from Design.label</th>
</tr>
</thead>
</table>

**labelPosition**

`labelPosition: "stacked" | "hidden" | "inline" (optional)`

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

<table>
<thead>
<tr>
<th>Inherited from Design.labelPosition</th>
</tr>
</thead>
</table>

**name**

`name: string (optional)`

<table>
<thead>
<tr>
<th>Inherited from Design.name</th>
</tr>
</thead>
</table>

**padding**

`padding: "none" | "small" | "std" (optional)`

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

| Inherited from Design.padding |
**type**

type: `ControlType` (optional)

The type of the control as a string.

Inherited from `Design.type`
Multi-Lookup metadata type.

**Hierarchy**

```
InputControlMetadata
 └ MultiLookupMetadata
```

**Index**

**Properties**

- BoundEntity
- BoundField
- Description
- Design
- Editable
- ExtType
- FilterContext
- FilterLocalOnly
- HelpText
- Hidden
- Id
- Label
- LookupPageId
- Mandatory
- Name
- NumSequence
- Order
- ReferenceAppId
- ReverseLookupRelation
- ShowPending
- Type

**Events**

- OnLookupPageCreate
- OnLookupPageCreated

**Properties**

**BoundEntity**

*BoundEntity*: string (optional)

The entity to which the control is bound.

Inherited from `ControlMetadata.BoundEntity`
BoundField
BoundField: string (optional)

Inherited from ControlMetadata.BoundField

Description
Description: string (optional)

Description of the control.

Inherited from ControlMetadata.Description

Design
Design: Design (optional)

Design object for the lookup page that is referenced by the LookupPageId.

Editable
Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

Inherited from ControlMetadata.Editable

ExtType
ExtType: ControlType (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

Inherited from ControlMetadata.ExtType

FilterContext
FilterContext: DataFilter (optional)

FilterLocalOnly
FilterLocalOnly: boolean (optional)

HelpText
HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

Inherited from ControlMetadata.HelpText

Hidden
Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

Inherited from ControlMetadata.Hidden

Id
Id: string (optional)
Identification string for a control.

Inherited from ControlMetadata.Id

Label
Label: string (optional)
Label for a control. For example, a control representing a person's first name might have a label "First Name".

Inherited from ControlMetadata.Label

LookupPageId
LookupPageId: string (optional)
Page that is hosted within the multi-lookup.

Mandatory
Mandatory: boolean (optional)
If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline.

Inherited from InputControlMetadata.Mandatory

Name
Name: string (optional)
Name of a control.

Inherited from ControlMetadata.Name

NumSequence
NumSequence: NumberSequenceConfig (optional)
Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Example:

```javascript
// hide number sequence reference page from users
metadataService.hideNavigation('numSeqReferencePage');

// parameters to be passed to 'numSequence' flag in configureControl
var configParam = {
    referencePageName: 'numSeqReferencePage',
    dataType: 'HcmPersonnelNumberId'
};

// setup 'PersonnelNumber' control as number sequence in the task 'add-worker'
metadataService.configureControl('add-worker', 'PersonnelNumber', { numSequence: configParam });
```

Inherited from InputControlMetadata.NumSequence

Order
Order: number (optional)
Number indicating the order in which a control will appear on a page.

- **ReferenceAppId**
  ReferenceAppId: string (optional)

- **ReverseLookupRelation**
  ReverseLookupRelation: boolean (optional)

- **ShowPending**
  ShowPending: boolean (optional)

- **Type**
  Type: `ControlType` (optional)

  String indicating the control type.

Inherited from `ControlMetadata.Order` and `ControlMetadata.Type`.

**Events**

- **OnLookupPageCreate**
  OnLookupPageCreate: function(args: any, multiLookup: any): void (optional)

- **OnLookupPageCreated**
  OnLookupPageCreated: function(args: any, multiLookup: any): void (optional)
Hierarchy

PageTarget
   └—— NavigationArgs

Index

Properties

- **label**
  label: string (optional)

- **options**
  options: any (optional)

- **params**
  params: PageOptions (optional)

  Inherited from PageTarget.params

- **replace**
  replace: boolean (optional)

  If set to true, removes current view firing navigation from navigation history stack.

- **to**
  to: string (optional)

  Inherited from PageTarget.to

- **url**
  url: string (optional)

  If provided, this link is directly opened.
Number Sequence Configuration type.

Properties
- **dataType**
  - dataType: string
  - The data type is used to lookup whether the number sequence is editable or not on the reference page.

- **referencePageName**
  - referencePageName: string
  - Page name of the page that defines if the num sequence is editable.
Page object type.

Hierarchy
Page

Index

Properties
- children
- dataLoadedInitially
- initialized
- metadata
- metadataLoaded
- pageContext
- pageFilter
- state
- syncError
- syncPending
- syncProcessing
- syncUnitEditable
- title

Methods
- canSubmit
- close
- getAction
- getActions
- getControl
- getDesign
- getEntityContext
- isEditable
- refreshData
- resume
- submit
- suspend

Events
- onClose
- onComplete
- onDataLoaded
- onInit
- onPreInit
- onRefresh
Properties

children
children: Control []
(Read-only) The list of all direct children controls of the page.

dataLoadedInitially
dataLoadedInitially: Promise <void>
(Read-only) A promise which resolves when the data has loaded for the first time. The promise continues to stay resolved for the rest of the page life.

initialized
initialized: boolean
(Read-only) True if the page instance has been initialized.

metadata
metadata: PageMetadata
(Read-only) The page metadata.

metadataLoaded
metadataLoaded: Promise <void>
(Read-only) A promise which resolves when the metadata has finished loading.

pageContext
pageContext: string
The current page context.

pageFilter
pageFilter: DataFilter
The current filter applied on the page.

state
state: PageState
(Read-only) The current state of the page.

syncError
syncError: boolean
(Read-only) True if the page's submission is in error state. This normally happens when the server rejects submissions due to validation errors. Refer to this topic for a detailed explanation of page data synchronization.

syncPending
syncPending: boolean
(Read-only) True if the page's submission is waiting to be synced. Refer to this topic for a detailed explanation of
page data synchronization.

**syncProcessing**

syncProcessing: boolean

(Read-only) True if the page instance is currently syncing its submission. Refer to [this topic](#) for a detailed explanation of page data synchronization.

**syncUnitEditable**

syncUnitEditable: boolean

(Read-only) True if it’s possible to edit a submission while it’s waiting to be synchronized. Refer to [this topic](#) for a detailed explanation of page data synchronization.

**title**

title: string

(Read-only) The title of the page.

## Methods

**canSubmit**

canSubmit(): boolean

Returns true if action page can be submitted and there are no validation/error messages.

**close**

close(): void

Dispose the page instance and all its lifecycle events.

**getAction**

getAction(actionName: string): [PageLink](#)

Get a page action by name. These include the actions in the action sheet/menu.

### Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>actionName</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

**getActions**

getActions(): [PageLink](#)[]

Get all page actions. These include the actions in the action sheet/menu.

**getControl**

getControl(controlName: string): [Control](#)

Get a page control by name. It recursively searches through all its children pages.

### Parameters
<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlName</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

**Returns** Control

**getDesign**

getDesign(): Design

Get the design object associated with the page.

**Returns** Design

**getEntityContext**

gentityContext(): EntityRef

Get current entity context.

**Returns** EntityRef

**isEditable**

isEditable(): boolean

Returns true if the page is an Action Page

**Returns** boolean

**refreshData**

refreshData(): Promise < void>

Force refresh page data.

**Returns** refreshData

**resume**

resume(): Promise < void>

Resume a temporarily suspended page.

**Returns** resume

**submit**

submit(): Promise < CompleteEventArgs>

Submit an Action.

**Returns** submit

**suspend**

suspend(): void

Temporarily suspend a page. For example, when the page is not the active view.

**Returns** suspend

**Events**

**onClose**

onClose: EventHook < null>

Event that is raised when a page is closed.

**onComplete**
onComplete: EventHook <any>

Event that is raised when an action is completed.

**onDataLoaded**

onDataLoaded: EventHook <any>

Event that fires when the page data has loaded. The event may be fired multiple times - every time new data is loaded.

**onInit**

onInit: EventHook <any>

Event that fires when a page instance has been initialized, and the metadata has been loaded.

**onPreInit**

onPreInit: EventHook <any>

Event that fires when a page instance has been initialized. This is fired before the metadata has been loaded.

**onRefresh**

onRefresh: EventHook <null>

Event that fires on forced page refresh, before new data has been loaded.

**onStateChange**

onStateChange: EventHook <null>

Event that fires when the page state changes.

**onSubmit**

onSubmit: EventHook <PageSubmitArgs>

Event that fires before an action is submitted. It can be intercepted for action validation/deferring. Refer to IPageSubmitArgs to know more about the available options.

**onSyncStatusChange**

onSyncStatusChange: EventHook <null>

Event that fires when the page sync status changes.
Represented by the data that is loaded into a page.

**Hierarchy**

**PageData**

**Index**

**Methods**

- `getControlValue`
- `setControlValue`

### Methods

**getControlValue**

```typescript
getControlValue(controlName: string): any
```

Gets the value of a control directly from the data set loaded in the page. The "value" is loosely defined across all different types of controls and generally indicates the primary single field value displayed or interacted with the control. Some complex controls (e.g., a lookup or a list) may not have a simple value and thus cannot be accessed via this API.

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlName</td>
<td>string</td>
<td>name of the control whose value is to be retrieved</td>
</tr>
</tbody>
</table>

**Returns any**

**setControlValue**

```typescript
setControlValue(controlName: string, value: any): any
```

Sets the value of a control directly into the data set loaded in the page. The "value" is loosely defined across all different types of controls and generally indicates the primary single field value displayed or interacted with the control. Some complex controls (e.g., a lookup or a list) may not have a simple value and thus cannot be accessed via this API.

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlName</td>
<td>string</td>
<td>Name of the control whose value is to be set</td>
</tr>
<tr>
<td>value</td>
<td>any</td>
<td>The value to be set</td>
</tr>
</tbody>
</table>

**Returns any**
PageLink type

11/24/2021 • 2 minutes to read • Edit Online

Pagelink control type. A pagelink is a control that navigates to another page.

Hierarchy

Control
  └ PageLink

Index

Properties

- container
- generic
- getDataSource
- hidden

Methods

- allowsNavigation
- applyDesign
- dataSource
- getCount
- getDesign
- getNavigationHandler
- isEditable
- metadata
- parent
- root
- showCount

Properties

container

container: boolean (optional)

True if the control is a container.

Inherited from Control.container

generic

generic: boolean (optional)

Inherited from Control.generic

getDataSource

getDataSource: function(): any
hidden
hidden: boolean
True if the control is hidden.

Inherited from Control.hidden

Methods
allowsNavigation
allowsNavigation(): boolean
Returns boolean

applyDesign
applyDesign(design: PageLinkDesign): void
Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

Overrides Control.applyDesign

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>design</td>
<td>PageLinkDesign</td>
<td>object containing design properties as keys</td>
</tr>
</tbody>
</table>

Returns void
dataContext
dataContext(): any

Inherited from Control.dataContext

Returns any
getCount
getCount(): number | string
Returns number | string

getDesign
getDesign(): Design
Returns the design object of this control.

Inherited from Control.getDesign

Returns Design
getNavigationHandler
getNavigationHandler(): NavigationArgs
Returns NavigationArgs
isEditable

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

Inherited from Control.isEditable

Returns boolean

metadata

metadata(): PageLinkMetadata

Returns the metadata object of this control.

Overrides Control.metadata

Returns PageLinkMetadata

parent

parent(): Control | Page

Returns the parent (control or page) of this control.

Inherited from Control.parent

Returns Control | Page

root

root(): Page

Returns the root form instance (page) of this control.

Inherited from Control.root

Returns Page

showCount

showCount(): boolean

Returns boolean
Pagelink design object type.

**Hierarchy**

Design

|-- PageLinkDesign

**Index**

**Properties**

- alignItems
- alignSelf
- background
- bindings
- border
- color
- excludedContext
- flexFlow
- flexSize
- fontSize
- fontWeight
- hideArrow
- icon
- justifyItems
- label
- labelPosition
- name
- navigation
- padding
- showCount
- style
- type

**Properties**

**alignItems**

alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to [this web page](#) for documentation on the "align-items" property.

| Inherited from Design.alignItems |

**alignSelf**
alignSelf: string (optional)

| Inherited from Design.alignSelf |

**background**

background: string (optional)

Sets the background color. If “theme” is used, then the color will match the app’s theme color.

- **blue:** #0078D7
- **pomegranate:** #911844
- **raspberry:** #8D398F
- **darkOrange:** #D24726
- **green:** #369F47
- **blueberry:** #17234E
- **grape:** #432158
- **lightBlue:** #5DB2FF
- **lightGreen:** #82BA00
- **pink:** #DC4FAD
- **teal:** #008299
- **mediumDarkBlue:** #004B8B
- **cordovan:** #570000
- **darkCordovan:** #380000
- **black:** #000000
- **lightGray:** #E8E8E8
- **light:** #FFF
- **dark:** #333333
- **neutral:** #555555
- **negative:** #D24726

**bindings**

bindings: any (optional)

| Inherited from Design.bindings |

**border**

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

| Inherited from Design.border |

**color**

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.
Consider setting the background color at the same time as necessary when setting this attribute. 

Note: if color is set to “theme”, the theme color of the app will be used.

The following colors are available:

- **blue**: 
  - #0078D7;
- **pomegranate**: 
  - #911844;
- **raspberry**: 
  - #8D398F;
- **darkOrange**: 
  - #D24726;
- **green**: 
  - #369F47;
- **blueberry**: 
  - #17234E;
- **grape**: 
  - #432158;
- **lightBlue**: 
  - #5DB2FF;
- **lightGreen**: 
  - #82BA00;
- **pink**: 
  - #DC4FAD;
- **teal**: 
  - #008299;
- **mediumDarkBlue**: 
  - #004B8B;
- **cordovan**: 
  - #570000;
- **darkCordovan**: 
  - #380000;
- **black**: 
  - #000000;
- **lightGray**: 
  - #e8e8e8;
- **light**: 
  - #fff;
- **dark**: 
  - #333333;

| Inherited from | Design.color |

**excludeContext**

*excludeContext: boolean (optional)*

**flexFlow**

*flexFlow: string (optional)*

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to [this web page](#) for documentation on the "flex-flow" property.

| Inherited from | Design.flexFlow |

**flexSize**

*flexSize: string (optional)*

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to [this web page](#) for documentation on the "flex" property.

| Inherited from | Design.flexSize |

**fontSize**

*fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)*
The proportional text size

| FontWeight
| fontWeight: "normal" | "bold" (optional)
| Normal or bold text.

| hideArrow
| hideArrow: boolean (optional)
| Allows an arrow ( > ) on a default styled navigation control to be hidden. By default, arrows are present in a navigation control. This property can only be added through the design object.

| icon
| icon: string (optional)
| Name of the icon that is displayed in the pagelink control. Here is a list of available icons.

| justifyItems
| justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)
| This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

| label
| label: string (optional)
| Inherited from Design.label

| labelPosition
| labelPosition: "stacked" | "hidden" | "inline" (optional)
| Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

| name
| name: string (optional)
| Inherited from Design.name

| navigation
| navigation: NavigationArgs (optional)
| Navigation object of the pagelink.

| padding
padding: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

| Inherited from Design.padding |

**showCount**

showCount: boolean (optional)

If true, shows a count of the records present in the list on the target page. This property is only suitable when the navigation target is a Page which contains a List control.

**style**

style: string (optional)

Determines the visual style of the pagelink control. Options:

- "inline": takes up the full width its container, with the label in-line with the icon
- "button": takes up only as much width as needed by the label, with the label below the icon

**type**

type: ControlType (optional)

The type of the control as a string.

| Inherited from Design.type |
Pagelink metadata type.

Hierarchy
ControlMetadata
  └— PageLinkMetadata

Index

Properties
- BoundEntity
- BoundField
- Description
- Editable
- ExcludeContext
- ExtType
- HelpText
- Hidden
- Icon
- IconSize
- Id
- Label
- Name
- Navigation
- Order
- ShowCount
- Style
- Target
- Type
- UseDataContext

Events
- OnNavigate

Properties

BoundEntity
BoundEntity: string (optional)

The entity to which the control is bound.

Inherited from ControlMetadata.BoundEntity

BoundField
BoundField: string (optional)
Inherited from `ControlMetadata.BoundField`

**Description**

Description: string (optional)

Description of the control.

Inherited from `ControlMetadata.Description`

**Editable**

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

Inherited from `ControlMetadata.Editable`

**ExcludeContext**

ExcludeContext: boolean (optional)

**ExtType**

ExtType: `ControlType` (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

Inherited from `ControlMetadata.ExtType`

**HelpText**

HelpText: string (optional)

The keyboard shortcut for a command. For example, 
"(Shift+F5)"

Inherited from `ControlMetadata.HelpText`

**Hidden**

Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

Inherited from `ControlMetadata.Hidden`

**Icon**

Icon: string (optional)

Name of the icon that is displayed in the page link control. Here is a list of available icons.

**IconSize**

IconSize: number (optional)

Determines the size of the icon that is displayed in the page link control.

**Id**

Id: string (optional)

Identification string for a control.

Inherited from `ControlMetadata.Id`

**Label**
Label: string (optional)
Label for a control. For example, a control representing a person's first name might have a label "First Name".

Inherited from ControlMetadata.Label

Name
Name: string (optional)
Name of a control.

Inherited from ControlMetadata.Name

Navigation
Navigation: NavigationArgs (optional)
Navigation object of the page link.

Order
Order: number (optional)
Number indicating the order in which a control will appear on a page.

Inherited from ControlMetadata.Order

ShowCount
ShowCount: boolean (optional)
If true, shows a count of the records present in the list on the target page. This property is only suitable when the navigation target is a Page which contains on a List control.

Style
Style: string (optional)
Determines the visual style of the page link control. Options:

- "inline": takes up the full width its container, with the label in-line with the icon
- "button": takes up only as much width as needed by the label, with the label below the icon

Target
Target: string (optional)
Name of the target action or page to navigate to when the page link is selected.

Type
Type: ControlType (optional)
String indicating the control type.

Inherited from ControlMetadata.Type

UseDataContext
UseDataContext: boolean (optional)

Events

OnNavigate
OnNavigate: function(navigation: NavigationArgs | string): any (optional)
An event that is triggered when the navigation is triggered.
Hierarchy
PageMetadata

Index

Properties
- Controls
- Design
- ID
- QuickSubmit
- SourcePageId
- SubmitButtonDesign
- Tasks
- Title

Events
- OnDataLoaded
- OnInit
- OnPreInit
- OnSubmit
- OnTaskSubmitted
- OnTaskSubmitting

Properties

Controls
Controls: ControlMetadata[] (optional)

Design
Design: Design (optional)

ID
ID: string (optional)

QuickSubmit
QuickSubmit: boolean (optional)

SourcePageId
SourcePageId: string (optional)

SubmitButtonDesign
SubmitButtonDesign: Design (optional)

Tasks
Tasks: PageMetadata[] (optional)
Title
Title: string (optional)

Events

OnDataLoaded
OnDataLoaded: function(sender: Page, dataWrapper: any): void (optional)

OnInit
OnInit: function(sender: Page): void (optional)

OnPreInit
OnPreInit: function(sender: Page): void (optional)

OnSubmit
OnSubmit: function(dataValues: any, args: any): void (optional)

OnTaskSubmitted
OnTaskSubmitted: function(taskHandle: any, taskOptions: any): any (optional)

OnTaskSubmitting
OnTaskSubmitting: function(taskOptions: any): any (optional)
PageOptions type

Properties

- appId
- design
- excludeContext
- filter
- filterLocalOnly
- pageContext
- pageld
- readOptions

appId
appId: string (optional)

design
design: Design (optional)

excludeContext
excludeContext: boolean (optional)

filter
filter: DataFilter (optional)

filterLocalOnly
filterLocalOnly: boolean (optional)

pageContext
pageContext: string (optional)

pageld
pageld: string (optional)

readOptions
readOptions: IReadOptions (optional)
Args supplied to the OnSubmit event of the page.

**Hierarchy**
PageSubmitArgs

**Index**

**Properties**
- `dataValues`
- `sender`

**Methods**
- `addMessage`
- `cancel`
- `getMessages`
- `isCancelled`
- `wait`

**Properties**

`dataValues`

Get the payload of the submit action.

`sender`

Get the sender page instance of the submit action.

**Methods**

`addMessage`

Add a validation/error message to be displayed.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>any</td>
<td></td>
</tr>
</tbody>
</table>

`cancel`
cancel(): any
Prevent the action from submitting.

Returns any

getMessages
getMessages(): string [ ]
Get all previously added messages

Returns string [ ]

isCancelled
isCancelled(): boolean
Check if the submit action is cancelled.

Returns boolean

wait
wait(promise: Promise <any>): any
Wait on a given promise before continuing with the submission. All promises attached via wait must resolve before the submit action is performed.

Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>promise</td>
<td>Promise &lt;any&gt;</td>
<td></td>
</tr>
</tbody>
</table>

Returns any
Hierarchy
PageTarget
  └ NavigationArgs

Index
Properties
  • params
  • to

Properties
params
defaults: PageOptions (optional)

to
to: string (optional)
Part control type. A part is a container control that contains only a page, allowing for a page to be embedded within a page.

**Hierarchy**

ContainerControl

| Part |

**Index**

**Properties**

- container
- generic
- getDataSource
- hidden

**Methods**

- applyDesign
- dataContext
- getControl
- getControlById
- getDesign
- getEntityRef
- getPartPage
- hasTarget
- isEditable
- metadata
- parent
- root

**Properties**

**container**

container: boolean

True if the control is a container.

- Inherited from ContainerControl.container
- Overrides Control.container

**generic**

generic: boolean (optional)

- Inherited from Control.generic
**getDataSource**
getDataSource: function(): any

Inherited from Control.getDataSource

**hidden**
hidden: boolean

True if the control is hidden.

Inherited from Control.hidden

**Methods**

**applyDesign**
applyDesign(IDesign: PartDesign): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

Overrides Control.applyDesign

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDesign</td>
<td>PartDesign</td>
<td>object containing design properties as keys</td>
</tr>
</tbody>
</table>

Returns void

**dataContext**

dataContext(): any

Inherited from Control.dataContext

Returns any

**getControl**

getControl(controlName: string): Control

Given the name of a control, returns the control instance.

Inherited from ContainerControl.getControl

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlName</td>
<td>string</td>
<td>control name</td>
</tr>
</tbody>
</table>

Returns Control

**getControlById**

getControlById(id: string): Control

Given the ID of a control, returns the control instance.
### Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>control ID</td>
</tr>
</tbody>
</table>

Returns **Control**

**getDesign**

getDesign(): Design

Returns the design object of this control.

Inherited from **Control.getDesign**

Returns **Design**

**getEntityRef**

gentityRef(): string

Gets value of entityRef binding to control.

Returns **string**

**getPartPage**

getPartPage(): Page

Gets the page of the part.

Returns **Page**

**hasTarget**

hasTarget(): boolean

Returns true if the part has a target page.

Returns **boolean**

**isEditable**

isEditable(): boolean

Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable. Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control’s edit-ability and its parent’s edit-ability is undefined.

Inherited from **Control.isEditable**

Returns **boolean**

**metadata**

metadata(): PartMetadata

Returns the metadata object of this control.

Overrides **ContainerControl.metadata**
**Returns** PartMetadata

**parent**

parent(): Control | Page

Returns the parent (control or page) of this control.

| Inherited from Control.parent |

**Returns** Control | Page

**root**

root(): Page

Returns the root form instance (page) of this control.

| Inherited from Control.root |

**Returns** Page
Part design object type.

**Hierarchy**

ContainerControlDesign  
├─ PartDesign

**Index**

**Properties**

- alignItems
- alignSelf
- allowScroll
- background
- bindings
- border
- color
- design
- flexGrow
- flexSize
- fontSize
- fontWeight
- itemBorder
- items
- justifyContents
- label
- labelPosition
- name
- padding
- target
- type

**Properties**

**alignItems**

alignItems: string (optional)

This property is an alias for the CSS property "align-items". Please refer to this web page for documentation on the "align-items" property.

Inherited from Design.alignItems

**alignSelf**

alignSelf: string (optional)
allowScroll
allowScroll: string (optional)

True if the container will allow scrolling when its items do not fit into the container’s available space. If a container has an item which may scroll, then set this property to false to prevent nested scrolling areas.

background
background: string (optional)

The background color of the container. Consider modifying the color attribute in the same container so that fonts overlaying the background color will appear appropriately. Note: if background is set to "theme", the theme color of the app will be used. The following colors are available:

- blue: #0078D7
- pomegranate: #911844
- raspberry: #8D398F
- darkOrange: #D24726
- green: #369F47
- blueberry: #17234E
- grape: #432158
- lightBlue: #5DB2FF
- lightGreen: #82BA00
- pink: #DC4FAD
- teal: #008299
- mediumDarkBlue: #004B8B
- cordovan: #570000
- darkCordovan: #380000
- black: #000000
- lightGray: #e8e8e8
- light: #fff
- dark: #333333

bindings
bindings: any (optional)

border
border: “none” | “solid” | “left” | “right” | “top” | “bottom” (optional)

The border behavior of a control. This property will not be inherited by the children.
**color**

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.

Consider setting the background color at the same time as necessary when setting this attribute.

Note: if color is set to “theme”, the theme color of the app will be used.

The following colors are available:

- **blue**: #0078D7
- **pomegranate**: #911844
- **raspberry**: #8D398F
- **darkOrange**: #D24726
- **green**: #369F47
- **blueberry**: #17234E
- **grape**: #432158
- **lightBlue**: #5DB2FF
- **lightGreen**: #82BAA0
- **pink**: #DC4FAD
- **teal**: #008299
- **mediumDarkBlue**: #004B8B
- **cordovan**: #570000
- **darkCordovan**: #380000
- **black**: #000000
- **lightGray**: #e8e8e8
- **light**: #fff
- **dark**: #333333

---

**design**

design: PartDesign (optional)

Design for the target page.

**flexFlow**

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property “flex-flow”. Please refer to [this web page](#) for documentation on the “flex-flow” property.

---

**flexSize**

flexSize: string (optional)
One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to [this web page](#) for documentation on the "flex" property.

| Inherited from Design.flexSize |

**fontSize**

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

| Inherited from Design.fontSize |

**fontWeight**

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

| Inherited from Design.fontWeight |

**itemBorder**

itemBorder: "solid" | "none" (optional)

If true, a border will appear around each row in the list. This property is equivalent to applying the border property individually to all items in the container.

| Inherited from ContainerControlDesign.itemBorder |

**items**

items: string | Design [ ] (optional)

An array containing the components to place inside of the container.

| Inherited from ContainerControlDesign.items |

**justifyItems**

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to [this web page](#) for documentation on the "justify-content" property.

| Inherited from Design.justifyItems |

**label**

label: string (optional)

| Inherited from Design.label |

**labelPosition**

labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

| Inherited from Design.labelPosition |
name
name: string (optional)

Inherited from Design.name

padding
padding: "none" | "small" | "std" (optional)

Allows specifying the component’s padding behavior. A component will inherit the padding behavior specified by its parent container components.

Inherited from Design.padding

target
target: PageTarget (optional)

Target page of the part.

type
type: ControlType (optional)

The type of the control as a string.

Inherited from Design.type
Part metadata type.

Hierarchy
ContainerControlMetadata
   └ PartMetadata

Index
Properties
  • BoundEntity
  • BoundField
  • Description
  • Design
  • Editable
  • ExtType
  • HelpText
  • Hidden
  • Id
  • Label
  • Name
  • Order
  • Target
  • Type

Properties

**BoundEntity**
BoundEntity: string (optional)
The entity to which the control is bound.

Inherited from ControlMetadata.BoundEntity

**BoundField**
BoundField: string (optional)

Inherited from ControlMetadata.BoundField

**Description**
Description: string (optional)

Description of the control.

Inherited from ControlMetadata.Description
**Design**

Design: `PartDesign` (optional)

Design for the target page.

**Editable**

Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control's edit-ability and its parent's edit-ability is undefined.

| Inherited from | ControlMetadata.Editable |

**ExtType**

ExtType: `ControlType` (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

| Inherited from | ControlMetadata.ExtType |

**HelpText**

HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

| Inherited from | ControlMetadata.HelpText |

**Hidden**

Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

| Inherited from | ControlMetadata.Hidden |

**Id**

Id: string (optional)

Identification string for a control.

| Inherited from | ControlMetadata.Id |

**Label**

Label: string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

| Inherited from | ControlMetadata.Label |

**Name**

Name: string (optional)

Name of a control.

| Inherited from | ControlMetadata.Name |
**Order**
Order: number (optional)

Number indicating the order in which a control will appear on a page.

| Inherited from | ControlMetadata.Order |

**Target**
Target: PageTarget (optional)

Target page of the part.

**Type**
Type: ControlType (optional)

String indicating the control type.

| Inherited from | ControlMetadata.Type |
Row controls are what make up a list. A list contains any number of row controls.

**Hierarchy**
Row

**Index**

**Properties**
- fieldList
- headerField
- hidden
- imageFields
- isSelected
- item
- template

**Methods**
- getControl
- getControlById
- getControlValueById
- getRowHeader
- getRowId
- hasImageField
- isEntityCreatedNew
- isEntityDeleted
- isEntityModified
- isEntitySyncPending
- select

**Properties**

**fieldList**
fieldList: Control []

**headerField**
headerField: Control

**hidden**
hidden: boolean
If true then the row will be hidden.

**imageFields**
imageFields: Control []
**isSelected**

isSelected: boolean

**item**

item: any

A container of rendered data.

**template**

template: Group

Group control that represents the template for a row.

## Methods

**getControl**

getControl(controlName: string): Control

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlName</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

**Returns** Control

**getControlById**

getControlById(id: string): Control

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>id can be valueKey or the displayKey of the list control metadata</td>
</tr>
</tbody>
</table>

**Returns** Control

**getControlValueById**

getControlValueById(id: string): string

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>id can be valueKey or the displayKey of the list control metadata</td>
</tr>
</tbody>
</table>

**Returns** string

**getRowHeader**

getRowHeader(): Control

**Returns** Control

**getRowId**

getRowId(): string

**Returns** string

**hasImageField**
hasImageField(): boolean

Returns true if the row has an image field.

Returns boolean

isEntityCreatedNew

isEntityCreatedNew(): boolean

Returns boolean

isEntityDeleted

isEntityDeleted(): boolean

Returns boolean

isEntityModified

isEntityModified(): boolean

Returns boolean

isEntitySyncPending

isEntitySyncPending(): boolean

Returns boolean

select

select(): any

Returns any
Value control type. This is the base class for single value controls.

Hierarchy
InputControl
  └ Value
    └ FileUploader
    └ HyperLink
    └ GenericValue

Index

Properties
- container
- generic
- getDataSource
- hidden

Methods
- applyDesign
- dataContext
- getDesign
- getValue
- isEditable
- metadata
- parent
- root
- setValue

Events
- onDataChanged

Properties

container
container: boolean (optional)
True if the control is a container.

| Inherited from Control.container |

generic
generic: boolean (optional)

| Inherited from Control.generic |
**getDataSource**

getDataSource: function(): any

Inherited from Control.getDataSource

**hidden**

hidden: boolean

True if the control is hidden.

Inherited from Control.hidden

## Methods

### applyDesign

applyDesign(design: Design): void

Applies given design to the design on the control. If a design already exists, the prototype chain of the design will be preserved.

Inherited from Control.applyDesign

### Parameters

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>design</td>
<td>Design</td>
<td>object containing design properties as keys</td>
</tr>
</tbody>
</table>

Returns void

### dataContext

dataContext(): any

Inherited from Control.dataContext

Returns any

### getDesign

getDesign(): Design

Returns the design object of this control.

Inherited from Control.getDesign

Returns Design

### getValue

getValue(): string

Returns the value of the control.

Returns string

### isEditable

isEditable(): boolean
Boolean indicating if the control is editable. Returns false when either the control or its parent is not editable.
Returns true when both the control and its parent are editable. Returns true when either the control or its parent is editable and the other is undefined. Returns undefined if both the control's edit-ability and its parent's edit-ability is undefined.

<table>
<thead>
<tr>
<th>Inherited from Control.isEditable</th>
</tr>
</thead>
</table>

**Returns** boolean

**metadata**

metadata(): ValueMetadata

Returns the metadata object of this control.

<table>
<thead>
<tr>
<th>Overrides InputControl.metadata</th>
</tr>
</thead>
</table>

**Returns** ValueMetadata

**parent**

parent(): Control | Page

Returns the parent (control or page) of this control.

<table>
<thead>
<tr>
<th>Inherited from Control.parent</th>
</tr>
</thead>
</table>

**Returns** Control | Page

**root**

root(): Page

Returns the root form instance (page) of this control.

<table>
<thead>
<tr>
<th>Inherited from Control.root</th>
</tr>
</thead>
</table>

**Returns** Page

**setValue**

setValue(value: string): void

Sets the value of the control.

**Parameters**

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

**Returns** void

**Events**

**onDataChanged**

onDataChanged: EventHook <null>

An event that is triggered when the input control's data changes.

| Inherited from InputControl.onDataChanged |
Value design object type.

**Hierarchy**

```
InputControlDesign
  └ ValueDesign
    └ FileUploaderDesign
    └ HyperLinkDesign
```

**Index**

**Properties**

- `alignItems`
- `alignSelf`
- `bindings`
- `border`
- `color`
- `flexFlow`
- `flexSize`
- `fontSize`
- `fontWeight`
- `justifyItems`
- `label`
- `labelPosition`
- `name`
- `padding`
- `type`

**Properties**

**alignItems**

`alignItems: string (optional)`

This property is an alias for the CSS property "align-items". Please refer to [this web page](#) for documentation on the "align-items" property.

Inherited from Design.alignItems

**alignSelf**

`alignSelf: string (optional)`

Inherited from Design.alignSelf

**bindings**

`bindings: any (optional)`
border

border: "none" | "solid" | "left" | "right" | "top" | "bottom" (optional)

The border behavior of a control. This property will not be inherited by the children.

color

color: string (optional)

The foreground color of the container. This will modify the color of all headers, items, labels, and icons within the container.

Consider setting the background color at the same time as necessary when setting this attribute.

Note: if color is set to "theme", the theme color of the app will be used.

The following colors are available:

- blue: #0078D7;
- pomegranate: #911844;
- raspberry: #8D398F;
- darkOrange: #D24726;
- green: #369F47;
- blueberry: #17234E;
- grape: #432158;
- lightBlue: #5DB2FF;
- lightGreen: #82BA00;
- pink: #DC4FAD;
- teal: #008299;
- mediumDarkBlue: #004B8B;
- cordovan: #570000;
- darkCordovan: #380000;
- black: #000000;
- lightGray: e8e8e8;
- light: #fff;
- dark: #333333;

flexFlow

flexFlow: string (optional)

Specifying this property makes the component a flex container component. This property is an alias for the CSS property "flex-flow". Please refer to this web page for documentation on the "flex-flow" property.
flexSize: string (optional)

One number or two numbers written as a string. For example, "(size to grow) [(size-to-shrink)]" to accommodate available space in the immediate flex container. This property is an alias for the CSS property "flex". Please refer to this web page for documentation on the "flex" property.

Inherited from Design.flexSize

fontSize

fontSize: "medium" | "xx-small" | "x-small" | "small" | "large" | "x-large" | "xx-large" (optional)

The proportional text size

Inherited from Design.fontSize

fontWeight

fontWeight: "normal" | "bold" (optional)

Normal or bold text.

Inherited from Design.fontWeight

justifyItems

justifyItems: "flex-start" | "flex-end" | "center" | "space-between" (optional)

This property is an alias for the CSS property "justify-content". Please refer to this web page for documentation on the "justify-content" property.

Inherited from Design.justifyItems

label

label: string (optional)

Inherited from Design.label

labelPosition

labelPosition: "stacked" | "hidden" | "inline" (optional)

Determines how a label is positioned, if at all. By default, labelPosition is set to stacked.

Inherited from Design.labelPosition

name

name: string (optional)

Inherited from Design.name

padding

padding: "none" | "small" | "std" (optional)

Allows specifying the component's padding behavior. A component will inherit the padding behavior specified by its parent container components.

Inherited from Design.padding
**type**

type: ControlType (optional)

The type of the control as a string.

Inherited from Design.type
ValueMetadata type

Value metadata type.

Hierarchy

InputControlMetadata
   └ ValueMetadata
      └ FileUploaderMetadata
      └ HyperLinkMetadata

Index

Properties

- BoundEntity
- BoundField
- Description
- Editable
- ExtType
- HelpText
- Hidden
- Id
- Label
- Mandatory
- Name
- NumSequence
- Order
- Type

Properties

**BoundEntity**

BoundEntity: string (optional)

The entity to which the control is bound.

Inherited from ControlMetadata.BoundEntity

**BoundField**

BoundField: string (optional)

Inherited from ControlMetadata.BoundField

**Description**

Description: string (optional)

Description of the control.
Editable
Editable: boolean (optional)

Boolean indicating if the control is editable. False when either the control or its parent is not editable. True when both the control and its parent are editable. True when either the control or its parent is editable and the other is undefined. Undefined if both the control’s edit-ability and its parent’s edit-ability is undefined.

ExtType
ExtType: ControlType (optional)

The extended control type. For example, a control of type Input might have an extended type of Barcode.

HelpText
HelpText: string (optional)

The keyboard shortcut for a command. For example, "(Shift+F5)"

Hidden
Hidden: boolean (optional)

Boolean indicating if the control is hidden or not.

Id
Id: string (optional)

Identification string for a control.

Label
Label: string (optional)

Label for a control. For example, a control representing a person's first name might have a label "First Name".

Mandatory
Mandatory: boolean (optional)

If set to true then input for the control is required for the task to be completed. Mandatory controls will have a red outline.
**Name**

Name: string (optional)

Name of a control.

Inherited from ControlMetadata.Name

**NumSequence**

NumSequence: NumberSequenceConfig (optional)

Used for auto detecting and changing visibility of the number sequence controls in the task or page, based on AX number sequence configuration, through extended business logic. Example:

```javascript
// hide number sequence reference page from users
metadataService.hideNavigation('numSeqReferencePage');

// parameters to be passed to 'numSequence' flag in configureControl
var configParam = {
    referencePageName: 'numSeqReferencePage',
    dataType: 'HcmPersonnelNumberId'
};

// setup 'PersonnelNumber' control as number sequence in the task 'add-worker'
metadataService.configureControl('add-worker', 'PersonnelNumber', { numSequence: configParam });
```

Inherited from InputControlMetadata.NumSequence

**Order**

Order: number (optional)

Number indicating the order in which a control will appear on a page.

Inherited from ControlMetadata.Order

**Type**

Type: ControlType (optional)

String indicating the control type.

Inherited from ControlMetadata.Type
Process automation enables simple scheduling of processes that will be run by the batch server. The process automation framework is a set of APIs that lets you implement process automation.

You should use only the public APIs to implement process automation, and you should follow these guidelines:

- Don't select from, insert into, or directly reference the process automation tables.
- Don't extend the framework or integrate your code with the classes.
- Don't subscribe to table events such as insert, update, and delete. Finance and Operations apps skip most of those events.
- If functionality that you require is missing, submit feature requests.

Microsoft plans to add features in the future. If you integrate too deeply with the process automation framework, your integration might break when those features are added.

Some of the examples for the process automation framework aren’t representative of release-quality code. As always, the expectation is that processes that are built by using the framework will follow all best practices and quality standards.

For more information about process automation, see Process automation.

### Definitions

<table>
<thead>
<tr>
<th>TERM</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poller</td>
<td>The poller is a system-critical batch process that runs every minute and invokes various subsystems of the process automation framework. It consults the schedule to determine which processes are ready to run, and then it invokes the runtime side of the framework to ensure that processes are run.</td>
</tr>
<tr>
<td>Scheduled process</td>
<td>A scheduled process is a process that is scheduled in the user interface (UI) by a user. Occurrences for these processes can be seen in a calendar view.</td>
</tr>
<tr>
<td>Background process</td>
<td>A background process is also known as a polled process. It’s a process that runs frequently, without requiring user input, and performs some background processing. Subledger transfer to the general ledger is an example.</td>
</tr>
<tr>
<td>Type</td>
<td>In this topic and related topics, the term type refers to ProcessScheduleType, as discussed in Type registration.</td>
</tr>
<tr>
<td>Series</td>
<td>Every process that has a registered type must have a series. Series for scheduled processes are created in the UI by users. Series for background processes are created through series registration. For more information, see Series registration.</td>
</tr>
<tr>
<td>Date and time</td>
<td>All framework dates are stored in Coordinated Universal Time (UTC) but shown in the user's preferred time zone.</td>
</tr>
</tbody>
</table>
## Tasks

Implementation of a process automation solution consists of a set of tasks, some of which are required and some of which are optional.

Most of the UI customizations aren't supported for background processes. The **Series** list page and logging of results and messages are supported.

<table>
<thead>
<tr>
<th>TASK</th>
<th>REQUIRED FOR A SCHEDULED PROCESS</th>
<th>REQUIRED FOR A BACKGROUND PROCESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type registration</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Series registration</td>
<td>Not supported</td>
<td>Yes</td>
</tr>
<tr>
<td>Process parameters</td>
<td>No</td>
<td>Not supported</td>
</tr>
<tr>
<td>User-configurable queries</td>
<td>No</td>
<td>Not supported</td>
</tr>
<tr>
<td>Run processes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Log results and messages</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Customize the user interface</td>
<td>No</td>
<td>See Customize the user interface.</td>
</tr>
</tbody>
</table>
To implement a process by using the process automation framework, you must first understand the concept of a type in the framework. A type is a unique process that is integrated with the batch framework and that uses the SysOperations framework, specifically the SysOperationServiceController class. The types are stored in the ProcessScheduleType table.

Here are a few examples of different types that are registered with the process automation framework:

- Vendor Payment Proposal (VendPaymProposalAutomationTypeRegistrationProvider class)
- Vendor Invoice Posting (VendInvoicePostProcessScheduleTypeRegistration class)
- Subledger transfer to general ledger (SubledgerJournalVoucherTransferServiceRegistration class)

If an existing process uses RunBaseBatch, consider wrapping it with SysOperationServiceController. The process automation framework doesn't support RunBaseBatch.

To register your type with the process automation framework, you must implement the ProcessScheduleITypeRegistration interface. This interface has a single method that returns an instance of ProcessScheduleTypeRegistrationItem.

If your process uses a feature flag, you must disable and enable the type as the feature is disabled and enabled, respectively.

- If you disable the feature flag for a type, the type doesn't appear in the user interface (UI). The scheduler won't schedule any occurrences or background processes of that type to run, and the runtime side of the process automation framework won't create any batch jobs for that type.
- If you enable the feature flag for a type, any occurrences or background processes that are scheduled to run in the past will be run immediately. Usually, this behavior is what you want. However, if it isn't what you want, consider disabling any series that is related to the type before you disable the feature flag.

Feature management has events that you can subscribe to. The method that you use to enable and disable types is ProcessScheduleTypeRegistration.enableOrDisableType.

Every time that a database synchronization runs, types and series are updated from their definitions in code. The only background settings that aren't updated are those that can be edited by the system admin. The process automation framework does this update by hooking SysSetup. The system admin can manually trigger this update through the settings at System administration > Setup > Initialize background.

The following example shows a process for a scheduled type. Note the following points:

- A background process doesn't have to set the Parameter tab list, because background processes don't support parameters.
- The type name isn't shown in the UI. The name should be a developer-created string such as VendorInvoiceBatchPosting. It's used internally as a key to reference your type for various purposes. It cannot be a label.
- The type name is used heavily with the SysPlugIn pattern. Most of the interfaces that are implemented for the process automation framework follow the SysPlugIn pattern and require that the type name be supplied by the ExportMetadataAttribute. In most cases in this pattern, the framework invokes the implementation of an interface only for the type that is being operated on, not for other types. Code examples in this topic and related topics follow this pattern.
using System.ComponentModel.Composition;

// The VendPaymProposalAutomationTypeRegistrationProvider class handles type registration for Vendor payment proposal automations.
[Export(identifierStr(Dynamics.AX.Application.ProcessScheduleITypeRegistration))]
public final class VendPaymProposalAutomationTypeRegistrationProvider
implements ProcessScheduleITypeRegistration
{
    private const LabelId Caption = literalStr('@CashManagement:VendPaymProposalAutomationTypeName');
    private const LabelId HelpText = literalStr('@CashManagement:VendPaymProposalAutomationSeriesWizardHelpText');
    private const MenuItemName SeriesFormMenuItemName = menuItemDisplayStr(VendPaymProposalAutomationCriteriaSeries);

    [Wrappable(false)]
    public ProcessScheduleTypeRegistrationItem getScheduleTypeRegistrationItem()
    {
        ProcessScheduleTypeRegistrationItem item = ProcessScheduleTypeRegistrationItem::construct();
        item.parmName(VendPaymProposalAutomationConstants::RegisteredTypeName);
        item.parmLabelId(Caption);
        item.parmScheduleType(ProcessScheduleProcessType::Scheduled);
        item.parmCompanyScope(ProcessScheduleTypeCompanyScope::SingleCompany);
        item.parmProcessAutomationTaskClassName(classStr(VendPaymProposalAutomationTask));
        item.parmParameterTabItemList(this.constructParameterTabItemList());
        item.parmIsEnabled(VendPaymProposalAutomationFeature::isEnabled());
        return item;
    }

    private List constructParameterTabItemList()
    {
        List criteriaTabItemList = new List(Types::Class);
        ProcessScheduleTypeRegistrationParameterTabItem criteriaTabItem = ProcessScheduleTypeRegistrationParameterTabItem::newFromMenuItem(SeriesFormMenuItemName);
        criteriaTabItem.parmCaption(Caption);
        criteriaTabItem.parmHelpText(HelpText);
        criteriaTabItemList.addEnd(criteriaTabItem);
        return criteriaTabItemList;
    }
}

using System.ComponentModel.Composition;
using Microsoft.Dynamics.ApplicationPlatform.FeatureExposure;

// The VendPaymProposalAutomationFeature class defines the Vendor Payment Proposal Automation feature.
[Export(identifierStr(Microsoft.Dynamics.ApplicationPlatform.FeatureExposure.IFeatureMetadata))]
internal final class VendPaymProposalAutomationFeature implements IFeatureMetadata,
IFeatureMetadataEnablementNotifiable
{
    private static VendPaymProposalAutomationFeature instance;
    private void new()
    {
    }

    private static void typeNew()
    {
        instance = new VendPaymProposalAutomationFeature();
    }

    [Hookable(false)]
    public static VendPaymProposalAutomationFeature instance()
    {
        return instance;
    }
}

The following example shows feature management for a process automation framework type.
return VendPaymProposalAutomationFeature::instance;
}

public FeatureLabelId label()
{
    return literalStr("@CashManagement:VendPaymProposalAutomationFeatureName");
}

public int module()
{
    return FeatureModuleV0::AccountsPayable;
}

public FeatureLabelId summary()
{
    return literalStr("@CashManagement:VendPaymProposalAutomationFeatureSummary");
}

public WebSiteURL learnMoreUrl()
{
    return "<your URL>";
}

public boolean isEnabledByDefault()
{
    return false;
}

public boolean canDisable()
{
    return true;
}

public void onEnabled()
{
    this.enableOrDisableRegisteredType(NoYes::Yes);
}

public void onDisabled()
{
    this.enableOrDisableRegisteredType(NoYes::No);
}

private void enableOrDisableRegisteredType(NoYes _isEnabled)
{
    ProcessScheduleTypeRegistration::enableOrDisableType(VendPaymProposalAutomationConstants::RegisteredTypeName,
        _isEnabled);
}

internal static boolean isEnabled()
{
    return Dynamics.AX.Application.FeatureStateProvider::isFeatureEnabled(VendPaymProposalAutomationFeature::instance());
}
**ProcessScheduleTypeRegistrationItem class**

The `ProcessScheduleTypeRegistrationItem` class is used as a part of type registration and contains information that is specific to your type.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>public ProcessScheduleTypeName parmName(ProcessScheduleTypeName _name = name)</td>
<td>The type name that is passed to the <code>item.parmName</code> method isn't shown to users. This value is a developer-defined string that is used when various events are invoked. It should be assigned as a constant, not as a label. <strong>Never use a label as a type name.</strong> Use names such as <code>VendPaymentProposal</code>.</td>
</tr>
<tr>
<td>public ProcessScheduleProcessType parmScheduleType(ProcessScheduleProcessType _scheduleType = scheduleType)</td>
<td>This method determines whether the process is scheduled or polled.</td>
</tr>
<tr>
<td>public ProcessScheduleCompanyScope parmCompanyScope(ProcessScheduleCompanyScope _companyScope = companyScope)</td>
<td>This method determines whether the process is a single-company process or a global process. A single-company process sets the company context in a batch, depending on the company that the user is in when a series is created. If the scope is global, the company context is ignored, and all jobs are in <code>dat</code>.</td>
</tr>
<tr>
<td>public LabelId parmLabelId(LabelId _labelId = labelId)</td>
<td>The label that this method returns is shown to users and represents the display name for your type. An example is <code>Vendor payment proposal</code>.</td>
</tr>
<tr>
<td>public className parmProcessAutomationTaskClassName(ClassName _processAutomationTaskClassName = processAutomationTaskClassName)</td>
<td>The class name that this method returns is the class name of the class that will implement the <code>ProcessAutomationTask</code> interface.</td>
</tr>
<tr>
<td>public NoYes parmIsEnabled(NoYes _isEnabled = isEnabled)</td>
<td>This method determines whether the type that you're registering is enabled by default. If your type is feature-managed, a default value is taken from the state of the feature. Be sure to implement the enabled and disabled feature management events. Enable and disable your type in the process automation framework in the appropriate way, by using <code>ProcessScheduleTypeRegistration.enableOrDisableType</code> method. Example code is shown earlier in this topic.</td>
</tr>
<tr>
<td>public List parmParameterTabItemList(List _parameterTabList = parameterTabList)</td>
<td>A process can have many parameter pages in the UI. These parameter pages contain parameters that are specific to the process. They are surfaced as form parts in the <strong>Create series</strong> wizard and edit occurrence dialog box. For each parameter page, an instance of the <code>ProcessScheduleTypeRegistrationParameterTabItem</code> class must be constructed and returned in the list. If the process doesn't require parameter pages, return <code>null</code>. For more information, see <code>Process parameters</code>.</td>
</tr>
</tbody>
</table>

**ProcessScheduleTypeRegistrationParameterTabItem class**

The `ProcessScheduleTypeRegistrationParameterTabItem` class represents information that is specific to a
The `Create series` wizard supports parameter pages by using embedded form parts. This value is the menu item that goes to the form part that is created by the team that owns the process.

### METHOD

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public MenuItemName parmMenuItemName(MenuItemName _menuItemName = menuItemName)</td>
<td>The caption on the parameter page.</td>
</tr>
<tr>
<td>public LabelId parmCaption(LabelId _caption = caption)</td>
<td>The caption on the parameter page.</td>
</tr>
<tr>
<td>public LabelId parmHelpText(LabelId _helpText = helpText)</td>
<td>The Help text for the parameter page.</td>
</tr>
<tr>
<td>public static ProcessScheduleTypeRegistrationParameterTabItem newFromMenuItem(MenuItemName _menuItemName)</td>
<td>Constructor that initializes the instance with the specified menu item name.</td>
</tr>
</tbody>
</table>
Every process must have a series. The concept of a series in process automation resembles the concept of a meeting series in Microsoft Outlook. However, a series in process automation is a series of scheduled runs of a process. For most scheduled process types, users create the series in the user interface (UI), and series registration never has to be implemented. However, if the process that is being implemented is a schedule series, you can skip this task.

Background processes typically create a series via code, by using series registration, because background processes tend to be "under the hood" processes that don't allow for user interaction. To create a series via code, you implement the `ProcessScheduleISeriesRegistration` interface. This interface contains a single method that returns an instance of `ProcessScheduleSeriesRegistrationItem`.

The process automation framework lets system admins change the default polling interval and unit for background processes.

Here is an example of a test series that is used to test the process automation framework.
using System.ComponentModel.Composition;

// Implements the ProcessScheduleISeriesRegistration to register the vendor invoice batch posting task 'Series' with the Process Automation.
[Export(identifierStr(Dynamics.AX.Application.ProcessScheduleISeriesRegistration))]
[ExportMetadata(classStr(ProcessScheduleISeriesRegistration),
  classStr(VendInvoicePostProcessScheduleSeriesRegistration))]
internal final class VendInvoicePostProcessScheduleSeriesRegistration implements ProcessScheduleISeriesRegistration
{

    [Hookable(false)]
    public ProcessScheduleSeriesRegistrationItem
    getProcessScheduleSeriesRegistrationItem()
    {
        ProcessScheduleSeriesRegistrationItem
        processScheduleSeriesRegistrationItem =
        ProcessScheduleSeriesRegistrationItem::construct();
        processScheduleSeriesRegistrationItem.parmDescription("@AccountsPayable:VendInvoicePostTaskFeatureSummary");
        processScheduleSeriesRegistrationItem.parmOwnerId(curUserId());
        processScheduleSeriesRegistrationItem.parmProcessScheduleSeriesPatternList(this.getSeriesPatternList());
        processScheduleSeriesRegistrationItem.parmSeriesName("@AccountsPayable:VendInvoicePostTaskFeatureLabel");
        processScheduleSeriesRegistrationItem.parmTypeName(VendInvoicePostTaskConstants::VendorInvoiceBatchPosting);
        return processScheduleSeriesRegistrationItem;
    }

    private List
    getSeriesPatternList()
    {
        ProcessScheduleSeriesPatternItem
        processScheduleSeriesPatternItem =
        ProcessScheduleSeriesPatternItem::construct();
        processScheduleSeriesPatternItem.parmUnit(ProcessScheduleUnit::Minute);
        processScheduleSeriesPatternItem.parmPollingInterval(VendParameters::pollingIntervalMinutes());
        List list = new List(Types::Class);
        list.addEnd(processScheduleSeriesPatternItem);
        return list;
    }
}

ProcessScheduleSeriesRegistrationItem class

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>public ProcessScheduleTypeName</td>
<td>The name of the type.</td>
</tr>
<tr>
<td>parmTypeName(ProcessScheduleTypeName _typeName = typeName)</td>
<td></td>
</tr>
<tr>
<td>public ProcessScheduleSeriesName</td>
<td>The name of the series. Be descriptive, so that the purpose of the series is clear from the name.</td>
</tr>
<tr>
<td>parmSeriesName(ProcessScheduleSeriesName _seriesName = seriesName)</td>
<td></td>
</tr>
<tr>
<td>public Description parmDescription(Description _description = description)</td>
<td>The description of the series.</td>
</tr>
<tr>
<td>public UserGroupId parmOwnerId(UserGroupId _ownerId = ownerId)</td>
<td>The user ID of the owner of the series.</td>
</tr>
</tbody>
</table>
The list of patterns for the series. Currently, only one pattern per series is supported. However, this limitation might change in the future. Insert an instance of the `ProcessScheduleSeriesPatternItem` class into the list.

### ProcessScheduleSeriesPatternItem class

When the pattern is configured, applicable fields are determined based on the unit. Not all methods that are defined in the following table work for all units. The methods that apply to units are defined in the tables later in this section. Other combinations will be ignored. For polled processes, only the unit and the polling interval are used. Other fields are ignored.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>public List</strong> parmProcessScheduleSeriesPatternList(List _seriesPatternList = seriesPatternList)**</td>
<td>The list of patterns for the series. Currently, only one pattern per series is supported. However, this limitation might change in the future. Insert an instance of the <code>ProcessScheduleSeriesPatternItem</code> class into the list.</td>
</tr>
<tr>
<td>public ProcessScheduleUnit parmUnit(ProcessScheduleUnit _unit = unit) **</td>
<td>The unit of time that the series runs in. The unit can be minutes or hours.</td>
</tr>
<tr>
<td>public ProcessScheduleInterval parmPollingInterval(ProcessScheduleInterval _pollingInterval = pollingInterval) **</td>
<td>For polled processes, this value is an integer that, together with the unit, defines how often the process runs.</td>
</tr>
<tr>
<td>public ProcessScheduleDateTime parmStartDate(ProcessScheduleDateTime _startDate = startDate) **</td>
<td>The start date of the series. The time should be set to the empty time.</td>
</tr>
<tr>
<td>public ProcessScheduleDateTime parmEndDate(ProcessScheduleDateTime _endDate = endDate) **</td>
<td>The end date of the series. The time should be set to the empty time.</td>
</tr>
<tr>
<td>public ProcessScheduleDateTime parmTime(ProcessScheduleDateTime _time = time) **</td>
<td>The time when the series should run. The date should be set to the empty date.</td>
</tr>
</tbody>
</table>

**Methods that are applicable to the week unit**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public NoYes parmOnSunday(NoYes _onSunday = onSunday) **</td>
<td>The days of the week that you want the process to run on by selecting the appropriate methods for the day of the week. For example, <code>parmOnMonday()</code> will run the job on a Monday.</td>
</tr>
</tbody>
</table>

**Methods that are applicable to the day unit**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public NoYes parmDoesRepeatEveryNumberOfDays(NoYes _doesRepeatEveryNumberOfDays = doesRepeatEveryNumberOfDays) **</td>
<td>Indicates whether the process should run every X number of days.</td>
</tr>
<tr>
<td>public int parmDailyRepeatInterval(int _dailyRepeatInterval = dailyRepeatInterval) **</td>
<td>Indicates the number of days.</td>
</tr>
<tr>
<td>public NoYes parmDoesRepeatEveryWeekDay(NoYes _doesRepeatEveryWeekDay = doesRepeatEveryWeekDay) **</td>
<td>Indicates whether the process should run every weekday.</td>
</tr>
</tbody>
</table>

**Methods that are applicable to the month unit**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
### Method

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>public NoYes parmDoesRepeatOnDayOfMonth(NoYes _doesRepeatOnDayOfMonth = doesRepeatOnDayOfMonth)</td>
<td>The process should run on a specific day of every month.</td>
</tr>
<tr>
<td>public Day parmMonthlyRepeatDayOfMonth(Day _monthlyRepeatDayOfMonth = monthlyRepeatDayOfMonth)</td>
<td>The day of month when the process should run.</td>
</tr>
</tbody>
</table>

### Modifying background processes

System admins can modify the polling interval and unit in the process automation framework. However, many background processes that currently exist have their own specific UI that is built to manage these changes. Microsoft provides a way to programmatically modify these values via the following APIs.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static ProcessScheduleSeriesPollingDetails</td>
<td>Gets the polling interval, the unit, and the next scheduled date/time for a</td>
</tr>
<tr>
<td>getPollingDetailsForSeries(ProcessScheduleTypeName _typeName,</td>
<td>polled process.</td>
</tr>
<tr>
<td>ProcessScheduleSeriesName _seriesName)</td>
<td></td>
</tr>
<tr>
<td>public static void</td>
<td>Enables changes to the polling interval, the unit, and the next scheduled</td>
</tr>
<tr>
<td>setPollingDetailsForSeries(ProcessScheduleTypeName _typeName,</td>
<td>date/time for a polled process.</td>
</tr>
<tr>
<td>ProcessScheduleSeriesName _seriesName,</td>
<td></td>
</tr>
<tr>
<td>ProcessScheduleSeriesPollingDetails _pollingDetails)</td>
<td></td>
</tr>
</tbody>
</table>

### Validating background process settings

In version 10.0.13, the process automation framework lets system admins modify background process settings via the **Edit background process** section. Some background processes have restrictions on the frequency of their runs. Microsoft has introduced an interface that a background process can implement. When this interface is invoked, it enables the background process to ensure that the unit and polling interval are within their allowed range.

The following example prevents this process from ever being run every minute or every hour. The process can run a maximum of one time per day. However, a process can implement the rules in such a way that more frequent runs are required.
using System.ComponentModel.Composition;

// Provider to validate background settings.
[Export(identifierStr(Dynamics.AX.Application.ProcessScheduleISeriesValidateBackgroundDialog))]
[ExportMetadata(extendedTypeStr(ProcessScheduleTypeName), 'ProcessAutomationExploder')]
internal final class ProcessScheduleExplodeAutomationBackgroundDialogValidationProvider implements ProcessScheduleISeriesValidateBackgroundDialog
{
    public boolean validateBackgroundProcessParameters(ProcessScheduleSeriesBackgroundValidationParameters _validationParameters)
    {
        ProcessScheduleUnit currentUnit = _validationParameters.parmUnit();
        if (currentUnit == ProcessScheduleUnit::Minute || currentUnit == ProcessScheduleUnit::Hour)
        {
            SysDictEnum enum = new SysDictEnum(enumNum(ProcessScheduleUnit));
            throw Error(strFmt("@ProcessAutomationFramework:ProcessScheduleExplodeProcessInvalidUnit", 
            enum.value2Label(ProcessScheduleUnit::Minute),
            enum.value2Label(ProcessScheduleUnit::Hour)));
        }
        return true;
    }
}

ProcessScheduleISeriesValidateBackgroundDialog interface
The ProcessScheduleISeriesValidateBackgroundDialog interface enables background processes to validate user input when users edit background settings via ProcessScheduleSeriesBackgroundDialog.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean validateBackgroundProcessParameters(ProcessScheduleSeriesBackgroundValidationParameters _validationParameters)</td>
<td>Implements any validation rules that you have for the process.</td>
</tr>
</tbody>
</table>

ProcessScheduleSeriesBackgroundValidationParameters class
The ProcessScheduleSeriesBackgroundValidationParameters class contains the validation parameters that are validated for the background processes that are being edited.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>public UserId parmOwnerId(UserId _ownerId = ownerId)</td>
<td>The owner of the process. It will be used when batch jobs are created, because batch jobs will be created under this user’s context.</td>
</tr>
<tr>
<td>public ProcessScheduleUnit parmUnit(ProcessScheduleUnit _unit = unit)</td>
<td>The unit of time.</td>
</tr>
<tr>
<td>public ProcessScheduleInterval parmPollingInterval(ProcessScheduleInterval _pollingInterval = pollingInterval)</td>
<td>The polling interval. It defines the number of units of time (as specified by parmUnit()) that the process should be run.</td>
</tr>
<tr>
<td>public ProcessScheduleDateTime parmPolledNextScheduledDateTime(ProcessScheduleDateTime _polledNextScheduledDateTime = polledNextScheduledDateTime)</td>
<td>The next scheduled run of the process in Coordinated Universal Time (UTC).</td>
</tr>
<tr>
<td>METHOD</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>public ProcessScheduleDateTime parmSleepFromTime(ProcessScheduleDateTime _polledSleepFromTime = polledSleepFromTime)</td>
<td>Specifies when the sleep should start. The process automation framework lets system admins put a process to sleep for a time range. The process isn't run during this time range, regardless of the setting of parmPolledNextScheduleDateTime(). This time range is a maximum of 16 hours and can span the date boundary.</td>
</tr>
<tr>
<td>public ProcessScheduleDateTime parmSleepToTime(ProcessScheduleDateTime _polledSleepToTime = polledSleepToTime)</td>
<td>Specifies when the sleep should end. The process automation framework lets system admins put a process to sleep for a time range. The process isn't run during this time range, regardless of the setting of parmPolledNextScheduleDateTime(). This time range is a maximum of 16 hours and can span the date boundary.</td>
</tr>
</tbody>
</table>
In most cases, a process must store custom parameters that are specific to its processes. For example, a process might require a date range or a customer number. You must create your own user interface (UI) and custom tables to show and store these parameters. If a type doesn’t have any parameters, you can skip this task.

When a user creates a series in the UI, the Create series wizard hosts multiple form parts, each of which contains a related set of parameters. The form parts for the process contain the UI that the user uses to enter the parameters. These form parts are built by the developer of the process and provided through type registration. A form part implements interfaces that let you initialize, validate, and write the custom parameters.

The custom parameter tables typically have two types of records:

- A template record that is bound to the series that serves as a template for all occurrences.
- A record that is specific to an occurrence and contains the parameters that will be used when that occurrence runs. Users can override the parameters for each occurrence as they require.

Parameter tables typically have one foreign key (RecId) to the ProcessScheduleSeries table and another foreign key (RecId) to the ProcessScheduleOccurrence table. The template record has a series foreign key, but it doesn’t have a foreign key to the occurrence. All other records have both foreign keys.

The following interfaces are used to maintain these parameters.

**ProcessScheduleParametersIInitialize interface**

The ProcessScheduleParametersIInitialize interface lets you initialize parameters when the user interacts with the UI of the process automation framework. The form part that is built for the wizard that shows process-specific parameters implements this interface.

**ProcessScheduleParametersIValidate interface**

The ProcessScheduleParametersIValidate interface lets you validate the parameters that the user enters in the form part.

**ProcessScheduleParametersIWrite interface**

The ProcessScheduleParametersIWrite interface lets you write the parameters to their custom parameter tables.

**Example**

In the following example, the three interfaces that were just described are used for a sample test process. In this example, the form part contains a single string that is known as a message.

```java
[Form]
public class ProcessScheduleSampleUptakeFirstFormPart
extends ProcessScheduleParametersFormPart
implements ProcessScheduleParametersIWrite, ProcessScheduleParametersIValidate,
ProcessScheduleParametersIInitialize
{
    private ProcessScheduleSchedulingContract schedulingContract;
```

public void setSchedulingContract(ProcessScheduleSchedulingContract _schedulingContract) {
  schedulingContract = _schedulingContract;
}

public void initializeForSeriesCreate() {
  String text = StringFmt("@ProcessAutomationFramework:ProcessScheduleSeriesTestTypeInitSeriesCreate", curExt());
  ProcessScheduleSampleUptakeParameters_Message.text(text);
}

public void initializeForSeriesUpdate() {
  ProcessScheduleSampleUptakeParameters parameters =
  ProcessScheduleSampleUptakeParameters::findForProcessScheduleSeries(schedulingContract.processScheduleSeries,
  true);
  ProcessScheduleSampleUptakeParameters_Message.text(parameters.Message);
}

public void initializeForOccurrenceCreate() {
  String text = StringFmt("@ProcessAutomationFramework:ProcessScheduleSeriesTestTypeInitOccurrenceCreate", curExt());
  ProcessScheduleSampleUptakeParameters_Message.text(text);
}

public void initializeForOccurrenceUpdate() {
  ProcessScheduleSampleUptakeParameters parameters =
  ProcessScheduleSampleUptakeParameters::findForProcessScheduleOccurrence(schedulingContract.processScheduleOccurrence);
  ProcessScheduleSampleUptakeParameters_Message.text(parameters.Message);
}

public void createScheduleSeries(ProcessScheduleSchedulingContract _schedulingContract) {
  ProcessScheduleSampleUptakeParameters sampleParameters;
  sampleParameters.Message = ProcessScheduleSampleUptakeParameters_Message.valueStr();
  sampleParameters.insert();
}

public void updateScheduleSeries(ProcessScheduleSchedulingContract _schedulingContract) {
  ProcessScheduleSampleUptakeParameters parameters =
  ProcessScheduleSampleUptakeParameters::findForProcessScheduleSeries(_schedulingContract.processScheduleSeries, true);
  if (parameters) {
    ttsbegin;
    parameters.Message = ProcessScheduleSampleUptakeParameters_Message.valueStr();
    parameters.update();
    ttscommit;
  }
}

public void createScheduledOccurrence(ProcessScheduleSchedulingContract _schedulingContract) {
  ProcessScheduleSampleUptakeParameters occurrenceParameters;
}
public void updateScheduledOccurrence(ProcessScheduleSchedulingContract _schedulingContract) {
    ProcessScheduleSampleUptakeParameters parameters = ProcessScheduleSampleUptakeParameters::findForProcessScheduleOccurrence(_schedulingContract.processScheduleOccurrence, true);

ttsbegin;

    parameters.Message = ProcessScheduleSampleUptakeParameters_Message.valueStr();

    if (parameters.RecId != 0) {
        parameters.update();
    } else {
        parameters.insert();
    }

ttscommit;
}

public boolean validate() {
    boolean isValid = true;
    if (ProcessScheduleSampleUptakeParameters_Message.valueStr() == '') {
        isValid = checkFailed("@ProcessAutomationFramework:ProcessScheduleSeriesTestTypeMessageWarning");
    }
    return isValid;
}

ProcessScheduleIDeleteOccurrence interface

Implement the ProcessScheduleIDeleteOccurrence interface to receive an event that indicates that a user or the system has deleted occurrences. The parameters that are related to that occurrence should be deleted.

This interface is invoked via SysPlugin for a specific type. Use the type name that was created when the type was registered.

Note that a Microsoft Azure SQL Database temp table is passed in so that you can do set-based deletes.
using System.ComponentModel.Composition;

// The VendPaymProposalAutomationOccurrenceDeleteProvider class is designed to handle
// deleting the appropriate VendPaymProposalAutomationCriteria records when ProcessScheduleOccurrence
// records are deleted.
[ExportMetadata(extendedTypeStr(ProcessScheduleTypeName), 'VendPaymProposalAutomation')]
[Export(identifierStr(Dynamics.AX.Application.ProcessScheduleIDeleteOccurrence))]
internal final class VendPaymProposalAutomationOccurrenceDeleteProvider
    implements ProcessScheduleIDeleteOccurrence
{
    private ProcessScheduleSeriesOccurrenceTmp occurrencesExplodedTmp;

    [Wrappable(false)]
    public void deleteOccurrences(ProcessScheduleSeriesOccurrenceTmp _occurrencesExplodedTmp)
    {
        this.initialize(_occurrencesExplodedTmp);
        this.deleteVendPaymProposalAutomationCriteria();
    }

    private void initialize(ProcessScheduleSeriesOccurrenceTmp _occurrencesExplodedTmp)
    {
        occurrencesExplodedTmp.linkPhysicalTableInstance(_occurrencesExplodedTmp);
    }

    private void deleteVendPaymProposalAutomationCriteria()
    {
        VendPaymProposalAutomationCriteria automationCriteria;
        automationCriteria.skipDeleteActions(true);
        automationCriteria.skipDataMethods(true);
        automationCriteria.skipAosValidation(true);
        automationCriteria.skipDatabaseLog(true);
        automationCriteria.skipEvents(true);
        delete from automationCriteria
            exists join occurrencesExplodedTmp
            where automationCriteria.ProcessScheduleOccurrence ==
                occurrencesExplodedTmp.ProcessScheduleOccurrence;
    }
}

ProcessScheduleIDeleteSeries interface

The ProcessScheduleIDeleteSeries interface resembles ProcessScheduleIDeleteOccurrence. The event is
invoked whenever a series is deleted. You should delete all parameter records for all occurrences. These records
include the series template record.

A SQL Database temp table is passed in so that you can do set-based deletes.
using System.ComponentModel.Composition;

// The VendPaymProposalAutomationSeriesDeleteProvider class is designed to handle
// deleting the appropriate VendPaymProposalAutomationCriteria records when ProcessScheduleSeries records are deleted.
[ExportMetadata(extendedTypeStr(ProcessScheduleTypeName), 'VendPaymProposalAutomation')]
[Export(identifierStr(Dynamics.AX.Application.ProcessScheduleIDeleteSeries))]
internal final class VendPaymProposalAutomationSeriesDeleteProvider
    implements ProcessScheduleIDeleteSeries
{

    [Wrappable(false)]
    public void deleteSeries(RefRecId _seriesRecId)
    {
        this.deleteVendPaymProposalAutomationCriteria(_seriesRecId);
    }

    private void deleteVendPaymProposalAutomationCriteria(RefRecId _seriesRecId)
    {
        VendPaymProposalAutomationCriteria automationCriteria;
        automationCriteria.skipDeleteActions(true);
        automationCriteria.skipDataMethods(true);
        automationCriteria.skipAosValidation(true);
        automationCriteria.skipDatabaseLog(true);
        automationCriteria.skipEvents(true);

        delete_from automationCriteria
        where automationCriteria.ProcessScheduleSeries == _seriesRecId;
    }
}

ProcessScheduleIExplodeOccurrences interface

When a user creates a new series through the UI, all the future occurrences are generated. Therefore, if the
series runs every day, the process automation framework creates an occurrence for every day. This action is
known as generating the series. The ProcessScheduleIExplodeOccurrences event is fired when the series is
generated. The series template record should be used as a template to create parameter records for each
occurrence in the parameter tables.

A SQL Database temp table is passed in so that you can do set-based creation of parameter records for optimal
performance.

In the following example, a parameter table stores a single parameter that is named Type. This parameter isn't
related to the process automation framework type but is specific to cash flow forecasting.
using System.ComponentModel.Composition;

// Provider for cash flow forecast automation generate occurrences.
[Export(identifierStr(Dynamics.AX.Application.ProcessScheduleIExplodeOccurrences))]
[ExportMetadata(extendedTypeStr(ProcessScheduleTypeName), 'LedgerCovTotalProcessAutomation')]
internal final class LedgerCovTotalProcessAutomationExplodeOccurrencesProvider
    implements ProcessScheduleIEplodeOccurrences
{
    private void new()
    {
    }

    [Wrappable(false)]
    public void explodeOccurrences(ProcessScheduleSeriesOccurrenceTmp _occurrencesExplodedTmp)
    {
        LedgerCovTotalProcessAutomationSchedulingParameters parameters;
        LedgerCovTotalProcessAutomationSchedulingParameters
            parametersSeriesRecord;

        insert_recordset parameters
        {
            Type,
            ProcessScheduleSeries,
            ProcessScheduleOccurrence
        }

        select Type, ProcessScheduleSeries from parametersSeriesRecord
            where parametersSeriesRecord.ProcessScheduleOccurrence == 0
            join ProcessScheduleOccurrence from _occurrencesExplodedTmp
            where _occurrencesExplodedTmp.ProcessScheduleSeries ==
            parametersSeriesRecord.ProcessScheduleSeries
                && _occurrencesExplodedTmp.TypeName ==
                LedgerCovTotalProcessAutomationConstants::RegisteredTypeName;
    }
}
This topic describes how to create configurable queries and use them with the process automation framework. If a process won't support user-configurable queries via the `SysQueryForm` form, you can skip this task.

The process automation framework provides limited support for custom queries via the `SysQueryForm` form. A custom query lets a user add custom criteria to limit how a process runs. The framework has logic to extract user-provided custom criteria and tables to store those criteria. The custom query criteria are stored for each occurrence of a given series and can be modified individually. The framework also provides an API to apply the custom criteria to the query that is used to run the process for each occurrence.

**NOTE**

When a user applies query criteria, the whole query object isn't saved. Instead the query criteria are saved individually, to allow for better support of query extensions. Therefore, extensions that are made to existing queries for existing query criteria should not cause breaking changes when this approach is used. In this case, a new extension should not require modification or re-creation of the query criteria for a saved series or occurrences. However, modification or re-creation of the query criteria is allowed.

### ProcessScheduleIQueryable interface

The `ProcessScheduleIQueryable` interface retrieves the original query (that is, the query as it was before the user modified it) and the user-modified query. These queries are used either to apply criteria when the process runs or to extract criteria when the user makes changes. These criteria are stored by the process automation framework.

This interface is accessed in the criteria form for a given implementation of process automation. For an example of access to this interface, see the `VendPaymProposalAutomationCriteria` form. That form also has a sample implementation of the `SysQueryForm` form.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public Query getOriginalQuery()</code></td>
<td>This method gets the original, unmodified query to use as a basis for comparison.</td>
</tr>
<tr>
<td><code>public QuerygetQueryForApplicationOrExtractionOfQueryCriteria()</code></td>
<td>This method gets the query that has been or will be modified, and that is used to apply or extract query criteria.</td>
</tr>
</tbody>
</table>

**NOTE**

The query that is used on the implementation of `ProcessScheduleIQueryable` must have the same structure as the query that is used during the run of the underlying process that is being automated. Any structural deviation that isn't additive in nature will cause runtime errors when the saved query criteria are applied at runtime. To ensure that the query structure remains the same, you should either use a designed query or use shared logic that builds up the query.

### ProcessScheduleQueryCriteriaApplicator class

The `ProcessScheduleQueryCriteriaApplicator` class is used to apply the saved query criteria for a given occurrence to the runtime instance of the query that is used when a process is run. This API must be called by an
uptaking process at a point in the run where the query is ready to accept the saved criteria. If a designed query or shared logic that builds up the query is used, this call can typically occur after the query has been correctly initialized. For an example that shows how this API is used, see the `CustVendCreatePaymJournal.constructFromAutomationExecutionContract` method.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static void applyCriteriaForOccurrenceExecution(Query _queryToApplyCriteria, RefRecId _scheduleOccurrenceRecId)</td>
<td></td>
</tr>
</tbody>
</table>
To run in the process automation framework, a process must implement the `ProcessAutomationTask` interface. The framework uses this interface to provide an instance of `ProcessScheduleWorkItem`. That instance contains information about the series, the occurrence that is being run (if applicable), and the execution ID. You must create the batch task that has to be run, and you must provide the task to the framework. The framework then creates the batch header and the tasks that are provided.

Polled processes don't have occurrences, because they might be run frequently, and those frequent runs will create many more occurrences than you want to track. Instead, you use a unique execution ID to track every run of a polled process. An execution ID is also assigned to scheduled processes.

The `getListOfWorkToBePerformed` method determines whether there is work that must be done. If there is, the method returns a list of batch-enabled classes that inherit from `SysOperationServiceController`. This method must be efficient and fast, because it's run in the context of the polling process. Therefore, you should not do any work in this method except check whether work must be done and create batch tasks for any work that must be done. It's OK if the method returns an empty list, because an empty list indicates that no work must be done. In this case, the process automation framework doesn't create any batch processes.

The process automation framework supports a list for those processes that do parallel processing and that want multiple batch tasks. If the process that is being run is an older process that implements `RunBaseBatch`, it can't be returned through the list. In this case, you have two options:

- Convert the process to `SysOperationServiceController`. For more information, see [SysOperations Framework](#). This option is recommended, if it's feasible.

- Wrap the legacy `RunBaseBatch` class with a new class that inherits from `SysOperationServiceController`. For an example, see [LedgerCovTotalProcessAutomationProcessor](#) in the Application Object Tree (AOT).

The following example shows an implementation of the `ProcessAutomationTask` interface.
using System.ComponentModel.Composition;

// The test class to test the process automation task engine.
[ExportMetadataAttribute(classStr(ProcessAutomationTask),
classStr(ProcessAutomationTaskTestImplementation))]
[ExportAttribute(identifierStr('Microsoft.Dynamics.AX.Application.ProcessAutomationTask'))]
internal final class ProcessAutomationTaskTestImplementation extends ProcessAutomationTask
{
    protected boolean isProcessAutomationEnabledForThisTask()
    {
        return true;
    }

    protected List getListOfWorkToBePerformed()
    {
        ProcessScheduleWorkItem scheduleWorkItemToExecute = this.parmProcessScheduleWorkItem();
        List taskList = new List(Types::Class);
        ProcessAutomationTestProcess testProcess = ProcessAutomationTestProcess::construct(scheduleWorkItemToExecute);
        taskList.addEnd(testProcess);
        return taskList;
    }

    // Gets and sets the batch job caption that will be used by the batch job scheduled to run this process.
    // Returns the batch caption that will be used for the batch job that will perform the task.
    protected BatchCaption batchJobCaption()
    {
    }
}

ProcessScheduleWorkItem class

The ProcessScheduleWorkItem class has many pieces of information and is designed to represent a process that must be run.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>public ProcessExecutionId parmExecutionId(ProcessExecutionId _executionId = executionId)</td>
<td>The execution ID is used to log errors. A polled process gets a new, unique execution ID every time that it's run. Because scheduled processes are only ever run one time for each occurrence, each occurrence only ever has one execution ID.</td>
</tr>
<tr>
<td>public RefRecId parmProcessScheduleOccurrenceRecId(RefRecId _scheduleOccurrenceRecId = scheduleOccurrenceRecId)</td>
<td>The occurrence that is being run. Use this method to reference occurrence-specific parameter information in parameter tables. Polled processes don't have a RecId value for an occurrence.</td>
</tr>
<tr>
<td>public RefRecId parmProcessScheduleSeriesPatternRecId(RefRecId _scheduleSeriesPatternRecId = scheduleSeriesPatternRecId)</td>
<td>The series pattern that the process is associated with. Currently, series can have only one pattern. All parameter records typically have a foreign key to this pattern.</td>
</tr>
<tr>
<td>public ProcessScheduleProcessType parmProcessScheduleType(ProcessScheduleProcessType _scheduleType = scheduleType)</td>
<td>The schedule type of the process: polled or scheduled.</td>
</tr>
<tr>
<td>public ProcessScheduleTypeName parmProcessScheduleTypeName(ProcessScheduleTypeName _scheduleTypeName = scheduleTypeName)</td>
<td>The type name that the process and series are associated with, such as VendPaymentProposal. This name is an internal developer name and isn't shown to the user.</td>
</tr>
<tr>
<td>METHOD</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>public List parmLegalEntityList(List _legalEntityList = legalEntityList)</td>
<td>The list of legal entities that the process will be run against. If the process is a global process, this list will be null. If the process is run against a single company, this list will contain a single company. Multiple companies aren't currently supported, but they might be supported in the future.</td>
</tr>
<tr>
<td>public Name getName()</td>
<td>Returns the occurrence name. If a type is polled, this method returns the series name.</td>
</tr>
<tr>
<td>public ProcessScheduleSeriesName parmSeriesName(ProcessScheduleSeriesName _seriesName = seriesName)</td>
<td>The name of the series.</td>
</tr>
<tr>
<td>public Name parmOccurrenceName(Name _occurrenceName = occurrenceName)</td>
<td>The occurrence name. If the process is a polled process, the value will be empty.</td>
</tr>
<tr>
<td>public List parmTaskList(List _taskList = taskList)</td>
<td>The list of batch tasks that the process automation framework should add to the batch. If this list is null, it's assumed that no work must be done, and nothing will be created in a batch.</td>
</tr>
<tr>
<td>public ProcessScheduleDateTime parmScheduledDateTime(ProcessScheduleDateTime _scheduledDateTime = scheduledDateTime)</td>
<td>The date and time when the process was scheduled to run. This date and time might differ from the actual date and time when the process runs.</td>
</tr>
<tr>
<td>public UserGroupId parmOwnerId(UserGroupId _ownerId = ownerId)</td>
<td>The owner of the occurrence that is being run.</td>
</tr>
<tr>
<td>public void initializeFromScheduleWorkItem(ProcessScheduleWorkItem _item)</td>
<td>Initializes an instance of ProcessScheduleWorkItem from another instance.</td>
</tr>
</tbody>
</table>
The process automation framework supports logging of results and messages. There are two reasons why a process should log results and messages:

- Results and the message log communicate the state of a process to the system admin or other roles that have access. It’s important that process results be monitored and seen by someone. If failures occur, they can be fixed, or an issue can be raised with the process owner.
- Results should communicate what the process did. For example, if the process posts vendor invoices, the results can show all the vendor invoices. In this case, each result will show the status and a link to the vendor invoice.

Results and messages are a multilevel logging system. A process has one or more results. Each result has one or more messages that are specific to it. A message is a composition child of the result. A result is typically something that the process is processing. For example, if the process posts vendor invoices, you log each vendor invoice as a result. Then, for each posting of each vendor invoice, you can log multiple messages that are associated with that result. If a vendor invoice is successfully posted, it’s OK to leave the message log blank, because the success of the operation should be obvious when users look at the result. If warnings occur, you should write them to the message log, even if posting was successful. These messages provide transparency, so that users can see what each process is doing and what the results are.

Both scheduled processes and polled processes support logging of results and messages. All processes should create a result. At a minimum, the result should communicate the fact that everything was successful. Processes that do work that affects user work and is visible to users should create more detailed results. For the vendor invoice posting example that was used earlier, users might want to see that their invoices have been posted. Results that don’t show that information can cause confusion, even if the information is available in other ways.

The following illustration shows the result view.

The following illustration shows the message view. To open this view, select View log in the Log column in the result view.
ProcessExecutionSourceLink table

The ProcessExecutionSourceLink table contains the results that are created while the processing is running. This table contains RefTableId and RefRecId fields. These fields are links to any source record in Microsoft SQL Server and are typically something that the process is processing. This table also contains Header and Message fields. The Header field will be shown as a column in the result grid. The message can be anything that you want it to be.

For example, if vendor payment proposal is creating a payment journal, the RefTableId value is the table ID of the LedgerJournalTable table. The RefRecId value is the RecId value of the LedgerJournalTable record that the payment journal created by the running process. In this case, you can set the Header field to the journal number. You can even make this value a jump reference, so that users can select the journal number to go directly to the payment journal. You can set the Message field to any message that you want to show, such as Payment journal created successfully.

If the process is processing many items (for example, if it’s posting many invoices), you can create a ProcessExecutionSourceLink record for each invoice. If the number of items that the process is processing is very large (in the millions), and users don’t have to see the details of each item, consider summarizing the items into batches. For example, the process for subledger transfer to the general ledger creates a ProcessExecutionSourceLink record for each transfer ID that is transferred to the general ledger, not for each voucher that is transferred.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static void insertSourceLinks(ProcessScheduleTypeName _typeName, ProcessExecutionSourceLinkTmp _tmp)</td>
<td>This method is new in version 10.0.14. It inserts many results into the ProcessExecutionSourceLink table by using set-based insert.</td>
</tr>
<tr>
<td>public static ProcessExecutionSourceLink insertSourceLink(ProcessExecutionSourceLinkItem _sourceLinkItem)</td>
<td>This method inserts a record into the ProcessExecutionSourceLink table.</td>
</tr>
</tbody>
</table>

ProcessExecutionSourceLinkItem class

Instantiate the ProcessExecutionSourceLinkItem class, and fill it with values that are required to correctly show your source item.
<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| public static ProcessExecutionSourceLinkItem newFromProcessScheduleWorkItemAndStatus(ProcessScheduleWorkItem _workItem, ProcessExecutionSourceStatus _status) | Use this constructor to create an instance of `ProcessExecutionSourceLinkItem`. This method correctly initializes many of the required fields from `ProcessScheduleWorkItem`.
| public static ProcessExecutionSourceLinkItem newFromProcessExecutionSourceLink(RefRecId _processExecutionSourceLinkRecId) | This method constructs an instance of `ProcessExecutionSourceLinkItem` and initializes the instance by using the specified record ID of a `ProcessExecutionSourceLink` record.
| public RefRecId parmSourceRecId(RefRecId _sourceRecId = sourceRecId) | Set the record ID of the source record. For example, this value might be the record ID of the vendor invoice header table.
| public RefTableId parmSourceTableId(RefTableId _sourceTableId = sourceTableId) | Set the table ID of the source table. For example, this value might be the table ID of the vendor invoice header table.
| public ProcessExecutionSourceLinkHeader parmHeader(ProcessExecutionSourceLinkHeader _header = header) | Set the value for the header field. For the vendor invoice posting example that was used earlier, this value might be the invoice number.
| public ProcessExecutionSourceLinkMessage parmMessage(ProcessExecutionSourceLinkMessage _message = message) | Set the message. For the vendor invoice posting example that was used earlier, this value might be **Posting successful**.
| public ProcessExecutionId parmExecutionId(ProcessExecutionId _executionId = executionId) | This method sets the execution ID. This value was provided via `ProcessScheduleWorkItem` in the implementation of the `ProcessAutomationTask` interface.

**ProcessExecutionMessageLog table**

The `ProcessExecutionMessageLog` table contains messages that are related to a single `ProcessExecutionSourceLink` record. You can write any type of message to this table. The message will then be shown to users.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| public static void insertMessages(ProcessScheduleTypeName _typeName, ProcessExecutionMessageLogTmp _tmp) | This method inserts messages into the `ProcessExecutionMessageLog` table by using a set-based insert.
| public static ProcessExecutionMessageLog insertMessage(ProcessExecutionMessageLogItem _errorLogItem) | This method inserts a message into the message log.

**ProcessExecutionMessageLogItem class**

The message log stores messages as both strings and label IDs. You don’t have to set both values. Label IDs are preferred, because they support translation of messages in the user interface (UI) for the message log. However, the messages are provided for backward compatibility with processes that don't support logging of label IDs.

Use the appropriate constructor for your scenario.
<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static ProcessExecutionMessageLogItem newFromProcessExecutionSourceLinkAndMessage(RefRecId _processExecutionSourceLinkRecId, Exception _exception, ProcessExecutionMessage _message)</td>
<td></td>
</tr>
<tr>
<td>public static ProcessExecutionMessageLogItem newFromProcessExecutionSourceLinkAndLabel(RefRecId _processExecutionSourceLinkRecId, Exception _exception, LabelId _labelId, container _labelParameters)</td>
<td></td>
</tr>
</tbody>
</table>
The process automation framework supports some customizations of the user interface (UI). Most of this topic is optional, because the framework provides default values for everything. The only exception is the `ProcessScheduleSeries` form. If you intend to show the `ProcessScheduleSeries` form for a specific product area, customizations are required so that the framework can show data that is specific to that product area.

**Weekly calendar view**

**ProcessScheduleIBuildOccurrenceCard interface**

The `ProcessScheduleIBuildOccurrenceCard` interface lets you customize the appearance of occurrence cards in the weekly calendar view. There is a static method on the interface for each status of an occurrence: `Scheduled`, `Waiting`, `Running`, `Successful`, `Failed`, and `Disabled`. You can create a customized occurrence card for each status value. Each of these methods returns an instance of `ProcessScheduleOccurrenceCard`.

The process automation framework provides a default implementation in the `ProcessScheduleOccurrenceCardBuilder` class. You inherit from this class and override the functionality as you require. You then register your derived class via the `SysPlugin` for your specific type. The registration process resembles the process for many of the plug-ins in the framework documentation.

An instance of `ProcessScheduleOccurrenceCardBuilderContract` is passed into each of the methods and can be used to retrieve information about the occurrence. The derived class can invoke the default implementation for each static method that returns the `ProcessScheduleOccurrenceCard` instance, modify whatever is required, and return it.

**ProcessScheduleOccurrenceCard class**

The `ProcessScheduleOccurrenceCard` class lets you customize the appearance of an occurrence card that is shown in the calendar view. The first two lines are controlled by the process automation framework and can’t be modified. In the following illustration, the subheader is the `Completed at` phrase, and the status message is the word `Completed` that has a blue background.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>public str parmSubHeader(str _subHeader = cardSubHeader)</code></td>
<td>The subheader, which is the third line of the occurrence card that is shown in the previous illustration.</td>
</tr>
</tbody>
</table>
### public str parmStatusMessage(str _statusMessage = statusMessage)
- The status message, which represents the status of the process and has a colored background.

### public ProcessExecutionOccurrenceCardStatusColor parmStatusColor(ProcessExecutionOccurrenceCardStatusColor _statusColor = statusColor)
- The color of the background for the status message.

### ProcessScheduleIShowOccurrenceCalendarView interface
The `ProcessScheduleIShowOccurrenceCalendarView` interface must be implemented by forms that will show the weekly calendar view. The `ProcessScheduleSeries` form is an example of a form that implements this interface.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProcessScheduleOccurrenceCalendarViewContract getProcessScheduleOccurrenceCalendarViewContract()</td>
<td>Return the contract that the weekly view will use to determine which types should be shown.</td>
</tr>
<tr>
<td>void refreshAfterChangeToCalendarView()</td>
<td>This value is a callback from the weekly view. It indicates that the parent form should be refreshed because of changes in the weekly calendar view.</td>
</tr>
</tbody>
</table>

### ProcessScheduleOccurrenceCalendarViewContract class
Use the `ProcessScheduleOccurrenceCalendarViewContract` class to limit the series that the weekly calendar view should show. For an example, see `ProcessScheduleSeries.getProcessScheduleOccurrenceCalendarViewcontract` in the Application Object Tree (AOT).

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static ProcessScheduleOccurrenceCalendarViewContract construct()</td>
<td>Use this constructor if the intention is to show occurrences for one to many types.</td>
</tr>
<tr>
<td>internal static ProcessScheduleOccurrenceCalendarViewContract newFromScheduleSeries(ProcessScheduleSeries _scheduleSeries)</td>
<td>Use this constructor if the intention is to show occurrences for a single series.</td>
</tr>
<tr>
<td>public void AddScheduleType(ProcessScheduleTypeName _scheduleTypeName)</td>
<td>If you aren't showing just one series, use this value to add the types that should be shown.</td>
</tr>
</tbody>
</table>

### ProcessScheduleOccurrenceCalendarViewRenderer class
Use the `ProcessScheduleOccurrenceCalendarViewRenderer` class to render the weekly calendar view in an existing form. A form part will be created and correctly initialized. An example of this class is used in the `ProcessScheduleSeries` form.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static ProcessScheduleICalendarView renderCalendarViewInFormControl(FormGroupControl _containingGroupControl)</td>
<td>This method renders the weekly calendar view in the specified form group control.</td>
</tr>
</tbody>
</table>

### Render interfaces
Several interfaces enable customization of the way that occurrence processes are rendered in the calendar view.
There is one interface for each status that a process can have:

- ProcessScheduleIRenderDisabledOccurrenceCard
- ProcessScheduleIRenderFailedOccurrenceCard
- ProcessScheduleIRenderRunningOccurrenceCard
- ProcessScheduleIRenderScheduledOccurrenceCard
- ProcessScheduleIRenderSuccessfulOccurrenceCard
- ProcessScheduleIRenderWaitingOccurrenceCard

All these interfaces follow the same pattern. An instance of ProcessScheduleOccurrenceCardRendering is sent to them. That instance is used to control how the occurrence card is rendered.

For an example, see the CustVendPaymProposalAutomationOccurrenceCardRenderer class in the AOT.

**ProcessScheduleOccurrenceCardRendering class**

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>public ProcessScheduleOccurrence getOccurrenceBeingRendered()</td>
<td>This method returns the occurrence that is being rendered on the occurrence card.</td>
</tr>
<tr>
<td>public ProcessExecutionExecutingInformation getOccurrenceExecutionInformation()</td>
<td>This method returns the running information for the occurrence. This information typically includes the results of the batch job, the start time, and the end time.</td>
</tr>
<tr>
<td>public void makeCardSubHeaderInvisible()</td>
<td>This method makes the card's subheader invisible. See the illustration earlier in this topic and the content below it to determine which line is the subheader.</td>
</tr>
<tr>
<td>public void makeCardButtonsInvisible()</td>
<td>This method specifies whether the Disable and Edit buttons on the occurrence card are invisible.</td>
</tr>
<tr>
<td>public void setColumnsOnOccurrenceCardDetailGroup(int _numberOfColumns)</td>
<td>This method enables the number of columns on the occurrence card to be customized. By default, there are two columns.</td>
</tr>
<tr>
<td>public FormButtonControl addButtonControl(FormControlName _buttonControlName)</td>
<td>This method enables a new button to be added to the occurrence card.</td>
</tr>
<tr>
<td>public FormStaticTextControl addStaticTextControl(FormControlName _staticTextControlName)</td>
<td>This method enables a static text control to be added to the occurrence card.</td>
</tr>
<tr>
<td>public FormStringControl addStringControl(FormControlName _stringControlName, LabelId _stringControlLabel)</td>
<td>This method adds a string control to the occurrence card.</td>
</tr>
</tbody>
</table>

**Series list page**

**ProcessScheduleISeriesFormController interface**

The Series list page uses the ProcessScheduleISeriesFormController controller to determine which types series will be shown for on the ProcessScheduleSeries list page. This class uses the SysPlugIn class. The menu item that you use to open the ProcessScheduleSeries form is used as the key to invoke the specified plug-in. This key enables each use of this form to customize which types are shown.
// Implementation of the ProcessScheduleISeriesFormController for the admin view of the process schedule Series form.
// This implementation will show series for all process types, both scheduled and polled, on the Series form.
[Export(identifierStr(Dynamics.AX.Application.ProcessScheduleISeriesFormController))]
[ExportMetadata(classStr(ProcessScheduleISeriesFormController), menuItemDisplayStr(ProcessScheduleSeriesAdmin))]
internal class ProcessScheduleSeriesFormAdminController implements ProcessScheduleISeriesFormController
{
    [Hookable(false)]
    public ProcessScheduleSeriesFormContract getSeriesFormContract()
    {
        return ProcessScheduleSeriesFormContract::newForAllScheduleTypes();
    }
}

**ProcessScheduleSeriesFormContract class**

The **ProcessScheduleSeriesFormContract** class is a contract that the series list page uses to determine which **ProcessScheduleType** is shown on it. This class can be used in a workspace to show series only for specific types that are related to that workspace.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>public void addScheduledScheduleType(ProcessScheduleTypeName _scheduleTypeName)</td>
<td>Add a specific scheduled type.</td>
</tr>
<tr>
<td>public void addPolledScheduleType(ProcessScheduleTypeName _scheduleTypeName)</td>
<td>Add a specific polled type.</td>
</tr>
</tbody>
</table>

**Results and messages**

**ProcessExecutionIResultsController interface**

The **ProcessExecutionIResultsController** interface lets you customize the results dialog box according to your process. It lets you set the column header label for the **Header** field in the results grid and make the value in the header column a hyperlink. This interface should be implemented by a class that is a plug-in. Here is a sample plug-in.
using System.ComponentModel.Composition;

[Export(identifierStr(Dynamics.AX.Application.ProcessExecutionResultsController))]
[ExportMetadata(classStr(ProcessExecutionResultsController), 'TestScheduledType')]
public final class ProcessExecutionSampleUptakeExecutionResultsController implements
ProcessExecutionIResultsController
{
    [Hookable(false)]
    public ProcessExecutionResultsDialogContract getResultsDialogContract()
    {
        ProcessExecutionResultsDialogContract contract = ProcessExecutionResultsDialogContract::construct();
        contract.parmSourceLinkHeaderLabel('Sample Header');
        contract.parmShouldSourceLinkHeaderBeLinkToSourceLinkDetails(true);
        return contract;
    }

    [Hookable(false)]
    public void openSourceLinkDetails(RefTableId _refTableId, RefRecId _refRecId)
    {
        if (_refTableId == tableNum(SystemParameters))
        {
            Args args = new Args();
            MenuFunction systemParametersMenuFunction = new
            MenuFunction(menuItemDisplayStr(SystemParameters), MenuItemType::Display);
            systemParametersMenuFunction.run(args);
        }
    }
}

### ProcessExecutionResultsDialogContract class

The **ProcessExecutionResultsDialogContract** class lets you customize the header column label and specify whether the data in the header column should be rendered as a hyperlink.

### ProcessExecutionMessageLogDialog class

The **ProcessExecutionMessageLogDialog** interface enables the message log to be opened in the context of something from the source domain. For example, the message log can be opened from the page for a posted vendor invoice to show the messages that were logged while the vendor invoice was being posted by a process that is enabled for the process automation framework. For this example, the posted vendor invoice page must
implement the `ProcessExecutionMessageLogDialog` interface. By using this interface, you don't have to build your own private results/messaging subsystems.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getContractForMessageLog()</code></td>
<td>This method returns an instance of <code>ProcessExecutionMessageLogContract</code>.</td>
</tr>
</tbody>
</table>

**ProcessExecutionMessageLogContract class**

The `ProcessExecutionMessageLogContract` contract lets you limit the message log to a specific item from the source domain. In the `ProcessExecutionSourceLink` table, there must be a record where the `RefRecId` and `RefTableId` values match the values that the contract sends.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>newForSourceRecord(ProcessScheduleTypeName _typeName, RefTableId _refTableId, RefRecId _refRecId, guid _executionId = emptyGuid())</code></td>
<td>This method initializes the contract by using the specified type name, <code>RefTableId</code> value, and <code>RefRecId</code> value. There should be a matching record in the <code>ProcessExecutionSourceLink</code> table. Background processes will have multiple execution IDs. Therefore, the optional parameter for the execution ID should be provided for background processes. For more information, see <a href="#">Type registration</a>.</td>
</tr>
</tbody>
</table>
The Regression suite automation tool (RSAT) significantly reduces the time and cost of user acceptance testing (UAT) of Finance and Operations apps. UAT is typically required before you take a Microsoft application update, or before you apply custom code and configurations to your production environment. RSAT lets functional power users record business tasks by using Task recorder and then convert the recordings into a suite of automated tests, without having to write source code. For more information about Task recorder, see Task recorder resources.

RSAT is fully integrated with Microsoft Azure DevOps for test execution, reporting, and investigation. Test parameters are decoupled from test steps and stored in Microsoft Excel files.

RSAT usage is described in these topics:

- Regression Suite Automation Tool (this topic)
- Regression Suite Automation Tool installation and configuration
- Run Regression Suite Automation Tool test cases
- Maintain test cases withinRegression suite automation tool
- Validate expected values
- Chain test cases
- Derived test cases
- Configure non-administrator users to use RSAT
- Upgrade the parameter files
- Regression Suite Automation Tool best practices
- Troubleshoot the Regression Suite Automation Tool

**Getting started videos**

These videos will help introduce RSAT and get you started.

**Use task recorder to create a test case for RSAT**

The How to use task recorder to create a test case for the Regression suite automation tool (RSAT) video (shown above) is included in the Finance and Operations playlist available on YouTube.

**Create a test plan in Azure DevOps to use with RSAT**

The How to create a test plan in Azure DevOps to use with the Regression suite automation tool (RSAT) video (shown above) is included in the Finance and Operations playlist available on YouTube.

**How to use RSAT**

The How to use the Regression suite automation tool (RSAT) video (shown above) is included in the Finance and Operations playlist available on YouTube.

**The improved Excel experience in RSAT 2.0**

The Improved Excel experience in RSAT 2.0 video (shown above) is included in the Finance and Operations playlist available on YouTube.
End-to-end flow

RSAT is part of the end-to-end flow described below. RSAT, Microsoft Dynamics Lifecycle Services (LCS), and Azure DevOps provide a set of tools for test case authoring (using Task recorder), distribution, configuration, execution, investigation, and reporting.

To learn more about this process, see [Create and automate user acceptance tests](#).

LCS, BPM, and Task Recordings

You aren't required to use the Business process modeler (BPM) tool in LCS. BPM is recommended if you want to enable the management and distribution of test libraries across projects and tenants. These capabilities are especially useful for Microsoft partners and independent software vendors (ISVs). BPM enables the distribution of test libraries as part of LCS solutions.

If you are not using BPM, you can manually create test cases in Azure DevOps and attach developer recording files to your Azure DevOps test cases. You can create developer recording files directly from the Task recorder pane.
You must name the developer recording file `Recording.xml` before attaching it to the Azure DevOps test case. Alternatively, you can name the recording file `-Test Case Title-.xml`, where `-Test Case Title-` is the DevOps title of the test case.

**Intended usage and test classification**

**Business cycle (business process) testing**

The Regression suite automation tool is intended to be used for business cycle tests and scenario tests (multiple component tests) that usually occur at the end of the development lifecycle. This is also referred to as user acceptance testing. Business cycle testing consists of a smaller number of test cases than component or unit testing. This is illustrated in the following graphic.

**Cloud POS**

In addition to testing processes recorded using the Finance and Operations Task recorder, RSAT also supports...
testing of Cloud POS processes in Dynamics 365 Commerce. For more information about RSAT with Cloud POS, see Test recorder and Regression suite automation tool for Cloud POS.

**Warehouse mobile app**
You can use RSAT in combination with the Warehouse App Task Validation Framework to automate the testing of warehouse processes. This Tech Talk is a good reference to get started.

**Unit and component testing**
For unit tests, we do not recommend that you use RSAT. Instead, use the SysTest framework and the build/test automation tools. For component tests, take advantage of the Acceptance test library resources (ATL). ATL is a library of X++ test helpers. When used with the SysTest framework, it offers the following benefits:

- Lets you create consistent test data.
- Increases the readability of test code.
- Provides improved discoverability of the methods that are used to create test data.
- Hides the complexity of setting up prerequisites.
- Supports high performance of test cases.

For more details, see Continuous delivery home page.

**Data integration testing**
Do not use RSAT for integration tests, instead rely on the data management framework (also known as DIXF). The Data task automation framework enables you to configure and automate the testing of your data integration scenarios.

**RSAT User interface overview**
RSAT 2.1 introduced a modern user interface that simplifies navigation through the main components of the app, including a Quick links tab, and quick navigation to DevOps test suites and test runs.

Use the left navigation pane to navigate between the test plan, settings, Cloud POS settings and the quick links page.

**Test Plan**
The Test plan tab is the main tab that allows you to interact with and execute test cases.

**Settings**
Select the Settings tab to configure RSAT settings. Use the top bar to navigate between general, optional and process settings. You do not need to save your settings, settings are automatically saved as soon as you navigate out of the settings page. You can also save your settings in an RSAT settings file or open an existing settings file.
Cloud POS Settings

Select the Cloud POS Settings tab to configure RSAT to execute Cloud POS test cases. You do not need to save your settings, settings will automatically be saved as soon as you navigate out of the settings page.

Useful links

The Links tab provides new functionality. Select the Links tab to quickly navigate to your Finance and Operations environment, Cloud POS, or go to useful Azure DevOps pages showing recent test runs, the last test run, and current test plan. There is also a link to the RSAT docs page.
Quick navigation to Azure DevOps

When working with your test plan, the **Open** button now provides 3 options.

- Open the selected test case in Azure DevOps.
- Open the selected test suite.
- Open the recent test runs.

This tab provides quick access to the most relevant pages in Azure DevOps.
This topic contains information about how to install and configure the Regression suite automation tool (RSAT).

**Prerequisites**

**Test environment (Prerequisite)**

Your test environment must be running Platform update 15 or newer. The Regression suite automation tool must have access to your test environment via a web browser.

**Excel**

You need Microsoft Excel installed to generate and edit test parameters.

**Azure DevOps (Prerequisite)**

You must have an Azure DevOps project to store and manage your test cases, test plans, and test case results. You will need an Azure DevOps Test Manager or Test Plans license. For example, if you have a Visual Studio Enterprise subscription, you already have a license to Test Plans. For more information, see Pricing for Azure DevOps Services or Pricing for Azure DevOps Server.

**Authentication Certificate**

RSAT is designed to be installed on any Windows 10 computer and connect remotely via a web browser to an environment.

To enable secure authentication, RSAT requires a certificate to be installed on the RSAT client computer. The RSAT settings dialog box allows you to automatically create and install the authentication certificate. You will also need to configure the virtual machine (VM) to trust the connection. Follow the instructions in the next sections to install and configure RSAT.

**Installation**

**Installer**

Download the .msi file from the Regression Suite Automation Tool Download to your machine and double-click it to run the installer.
NOTE
If you're using Azure DevOps Server, download and install version 1.210.48249.4 or later.

Selenium and Browser Drivers
RSAT requires Selenium and web browser driver libraries. RSAT will prompt you if needed libraries are missing and will automatically install them for you. Select Yes when you see the following (or similar) messages.

RSAT uses Selenium 3.13.1. The web driver library and browser-specific drivers are downloaded to C:\Program Files (x86)\Regression Suite Automation Tool\Common\External\Selenium.

Configuration
1. Open RSAT from your desktop.

2. Select the Settings tab on the upper left to configure RSAT.
General settings

These settings are required.

**Azure DevOps (General settings)**

Configure your connection to the Azure DevOps project and test plan.

- **Azure DevOps URL** - This is the URL of your Azure DevOps organization. For example, https://yourAzureDevOpsUrlHere.visualStudio.com.

  **NOTE**

  If you're using Azure DevOps Server, add /DefaultCollection to the end of your Azure DevOps URL.

- **Access Token** - The access token that allows the tool to connect to Azure DevOps. You need to create a personal access token or use an existing one that you have saved. It is recommended you create this with scope selected as Full Access. For more information, see Authenticate access with personal access tokens.

- **Project Name** - The name of your Azure DevOps project. RSAT will automatically detect project names and test plans available based the Azure DevOps URL specified. You can then select the Test Project and Test Plan.

- **Test Plan** - The Azure DevOps test plan that contains your test cases. For more information, see Create test plans and test suites.

Select Test Connection to test your connection to Azure DevOps.

**Test environment (General settings)**

Configure your connection to the test environment.
- **Hostname** – The hostname of the test environment, such as myhost.cloudax.dynamics.com. Don't include the https:// or http:// prefix.

- **SOAP Hostname** – The SOAP hostname of the test environment.
  - For demo and development environments (also known as one-box environments), add a `soap` suffix to the hostname. For example, if your hostname is `myhost.cloudax.dynamics.com`, use `myhost.soap.cloudax.dynamics.com` as the SOAP hostname.
  - If you don't know the SOAP hostname of your test environment, you can find it in the web.config file for the AOS server in `Infrastructure.SoapServicesUrl`.
  - If your test environment is a user acceptance testing (UAT) or higher-tier sandbox environment that has no Remote Desktop access, the SOAP hostname is equal to the hostname.

- **Admin User Name** – The email address of an admin user in the test environment. The admin user name must be the email address of a user who belongs to the System Administrator role on the Finance and Operations test environment that RSAT is connecting to. The user account (email address) must also belong to the same tenant as the test environment. For example, if your test environment's default tenant is contoso.com, the admin user must end with @contoso.com.

- **Thumbprint** – The thumbprint of the authentication certificate that you're using. If you don't have Remote Desktop Protocol (RDP) access to your environment, follow the steps lower in this article to download the certificate from Lifecycle Services and paste the thumbprint here. Otherwise, if you do have RDP access to the environment, follow these steps to generate a self-signed certificate.
  1. Select **New** to create and install a new authentication certificate. When prompted, place the .cer file somewhere so you have it saved for your records.
  2. When the process completes, the new certification is installed in the local machine's trusted root store.

```plaintext
Message

The certificate has been successfully created and added to the Trusted Root
Store.

OK
```

3. The thumbprint of the newly created certificate is automatically inserted on this form. Copy this thumbprint, you will use it in the next section to configure the AOS to trust the connection.

- **Company name** – Specify a company name to use as your default company during creation of Excel parameters files. It can be changed later by editing an Excel file.

Run setting
Configure your local settings.

- **Working directory** - Folder location for storing test automation files, including Excel test data files. For example: `C:\Temp\RegressionTool`.
- **Default browser** - Select the browser to use for test execution. RSAT supports (the new) Microsoft Edge, Microsoft Internet Explorer, and Google Chrome. We recommend Microsoft Edge, which you can download from [Introducing the new Microsoft Edge](https://www.microsoft.com/en-us/surface/windows-10-edge).

Select **Ok** to apply your settings and close the dialog box. Select **Cancel** to cancel your changes and close the dialog. The **Save As** and **Open** buttons allow you to save your settings for reuse later. Select **Save As** to save your current settings into a configuration file on your computer. Select **Open** to restore your settings from a configuration file.

Optional settings
Select the **Optional** tab to configure optional settings.

- **Test Run Prefix** – RSAT reports test run results to Azure DevOps. Test runs are named using the following convention: `<Run ID> <Prefix> <Test Suite>`. Use this setting to set the `<Prefix>`.

- **Test Run Timeout** – The time-out (in minutes) of a test run. All active windows are closed and pending test cases fail when this time-out is reached.

- **Test Action Timeout** – The time-out (in minutes) of individual test steps. When a test step times out, the test case fails.

- **Pause between steps** – The number of seconds to pause between test steps during automated execution of a test case. The default value is 0 (zero). Set this value to force a pause during test execution, for auditing or investigative purposes. You can also specify a pause for an individual test case by changing the **Pause between steps (Seconds)** parameter on the **General** tab of the Excel parameter file for the test case.

- **Fail test on first validation error** – By default, if a test case has multiple validation steps, and there is a validation failure, the test case stops running when the first failure occurs. The test case is then marked as failed. If you want test cases to continue to run until all validations are completed, clear this option. The test case can then evaluate all validations.

- **Fail test on Infolog error** - Check this option to force test cases to fail when an error is encountered in the Finance and Operations Infolog during test case execution.

- **Abort test suite execution on failure** – By default, a test suite run continues even if one of the test cases fails. If you check this setting, the test run is aborted if a test case fails. All the remaining test cases will have a status of **Not Executed**.

- **Enable local file validation rules** - Check this setting to validate whether your test cases are ready for execution. See Validate readiness of test automation files for more details.

- **Enable upload to Azure DevOps** - To prevent accidental upload to Azure DevOps (therefore overriding project-wide recordings and automation files), you can uncheck this setting. This is especially useful when RSAT is deployed on a client machine for execution purposes only, and you want to prevent users from making permanent changes to the test cases.

- **Cloud provider** – Select the provider of the cloud tenant of your test environment. Supported providers are **Global** (Public cloud) and **China** (Sovereign cloud).

  **IMPORTANT**
  
The **Cloud provider** setting is required, and the selected value must be **China** if your Finance and Operations apps were deployed in 21Vianet.

**Configure the test environment to trust the connection**

*If your AOS allows for Remote Desktop connections*

After creating the certificate, configure AOS to trust the test automation connection. On a multi-AOS environment, repeat the following steps for all AOS machines.

1. Open a Remote Desktop connection to the AOS machine.

2. Open IIS and find AOSService in the list of sites.
3. Right-click AOSService, then select Explore.

4. Open and find the file wif.config.

5. Update the wif.config file by adding a new authority entry, as shown in the following example. Use 127.0.0.1 for the authority name and paste your certificate thumbprint.

If you have no Remote Desktop access to the server

In cases where your Remote Desktop Protocol (RDP) access is removed, such as Microsoft-managed or self-service type sandboxes, Microsoft will generate the certificate for your environment and have it pre-configured. Follow these steps to retrieve the RSAT certificate and use it using the LCS user interface. For automation, there is information on the Fetch an environment's RSAT certificate in a zip file API reference page.

1. Under Maintain on your environment details page in Lifecycle Services you'll see two new options.
   - Download RSAT certificate
   - Regenerate RSAT certificate
Use the **Download** button to retrieve the certificate bundle as a .zip file.

2. You'll receive a warning that a clear-text password will be displayed on your screen. You will need the password in subsequent steps. Select **Yes** to continue.

3. Copy the clear-text password for later use. You'll see the .zip file has been downloaded. Inside the .zip file is a certificate (.cer) and a personal information exchange (.pfx) file. Unzip the file.

4. Install the certificate in the local machine's trusted root store:
   - Double-click the certificate (.cer) to open it, and then select **Install Certificate**.
   - Select *local machine*, and then browse to the *Trusted Root Certification Authorities* store to install it in the trusted root store.

5. Install the pfx file in the local machine's personal store:
   - Double-click the personal information exchange (.pfx) file to open it, and select **Local Machine**.
   - Enter the password saved in step 2, and browse to the *Personal* store.

6. Double-click the certificate file to open it. Browse to the **Details** tab, and scroll down until you see the **Thumbprint** section. Select **Thumbprint**, and note the ID in the text box. Select or paste this thumbprint in RSAT settings.
You can now run your tests against the environment using this certificate. The certificate will be autorotated by Microsoft before it expires, at which time you will need to download a new version of this certificate starting from step 1 above. For self-service environments, this will be rotated every 60 days during a downtime window that is closest to the expiry. These downtime windows include customer initiated package deployment, and database movement operations that target the environment.

Manual configuration of authentication certificates

Optionally, you can manually configure the RSAT authentication certificate.

If you are not familiar with this process, get help from your system administrator. Make sure you have Windows Kits installed on your machine. If you do not have Windows Kits installed on your machine, you can download the Windows 10 SDK from Windows 10 SDK. You will need these components for the steps described in this document.

- Windows SDK Signing Tools for Desktop Apps
- Windows SDK for UWP-Managed Apps.

Generate the certificate

You must generate the certificate file on the RSAT client computer. The certificate must be generated on the same computer that the test tool is running on. To generate the certificate file, follow these steps:

1. Create the C:\Temp folder if it does not already exist on your computer.

2. Open a command-line window as Administrator.

3. Go to the folder where you installed the Windows SDK. Your exact folder may be different, depending on where you have installed the windows SDK. You can also use Windows Kits 8.1.

   ```
   cd c:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64
   ```

4. Run the following command. When you are prompted to enter a private key password, enter None.

   ```
   makecert.exe -n "CN=127.0.0.1" -ss My -sr LocalMachine -a sha256 -len 2048 -cy end -r -eku 1.3.6.1.5.5.7.3.1 c:\temp\authCert.cer
   ```
Install the certificate to the Trusted Root

To install the certificate, follow these steps:

1. Double-click authCert.cer to install the certificate.

2. Select Install Certificate.

3. Select Local Machine > Place all certificates in the following Store > Browse > Trusted Root Certification Authorities and select Next through each screen.

4. Leave the Password field blank.

5. In the Certificate dialog box, browse to Details and look for Thumbprint.

6. Copy and save the thumbprint. You will need it to configure the AOS as described earlier in this topic.
This topic explains how to load test cases from Azure DevOps, generate automation files, modify test parameters, run and investigate results, and save your work back to Azure DevOps.

Load test cases and create automation files

In RSAT, select the **Test Plans** tab and then select **Load** to download test cases and test case automation files. All test cases (and their corresponding attachments) belonging to the test plan specified in the **Settings** tab are downloaded to the local working directory.

Test cases are organized by test suites under a common test plan. These are test suites you created in your Azure DevOps project. Using this tool, you can work with one test suite at a time.

If the tool fails to load any test case, verify that your test plan in Azure DevOps is properly created and contains the desired test suites and test cases.

If this is the first time you are using this test plan, the **Parameters File** column will be blank. You must create test automation files for your test cases.

A test case requires the following attachments for successful execution:

- **A recording file**: This is the recording file created by the Finance and Operations Task recorder. It defines the steps of your test case. It is typically named `recording.xml` but you can also name it to match the test case title in Azure DevOps. It is attached to the test case in Azure DevOps and downloaded into the attachment folder of the local working directory of the test case.

- **Test automation files** consisting of a **test parameter file** (Microsoft Excel file) containing configurable test case parameters and **test execution files**: These files are generated by RSAT to enable automated execution of the test recording. Filenames are suffixed by `_Base.cs`, `_Base.xml`, and `_Base.dll`.

When you select **New**, test automation files are generated in your working directory. The Excel test parameter files will appear on the grid under **Parameters File**.
You can also generate test execution files only, without overwriting your parameter files. Select New > Generate Execution Files to regenerate only execution files and leave Excel files unaffected.

You must generate test execution files when you install a new version of the tool, and when you modify or load a new version of the recording file. In this way, you update your execution files but also preserve the test parameter files.

Modify test parameters

This section describes how to modify Excel files to specify input and validation parameters for your test run. Select one or more test cases to modify, and then select the Parameters button (Microsoft Excel symbol) on the toolbar. An Excel window opens for each test case that you selected. Alternatively, you can open the Excel files directly from the working directory.

In addition to the General tab, the Excel parameter file contains a MessageValidation tab and a TestCaseSteps tab.

Select the TestCaseSteps tab to configure input and validation parameters of your test case. Input and validation parameters are placed directly next to their corresponding test case step, enabling test authors with context and a simple experience. When you modify parameters, it is clear what steps of the test case you are affecting. You can enter values or formulas in context. Color coding differentiates input parameters from validation steps.
Reusable variables that are copied while recording the test case are also shown in context of the test case step. You can easily locate a variable and copy it to use in subsequent steps and formulas. For more information, see Copy variables to chain test cases.

Save the Excel files when you are done making edits.

**Run a test as a specific user**

By default, tests are executed using the admin role. If you want to run the test as a specific security role, specify the email address of a user under the Test User parameter in the General tab of the Excel parameter file. The Test User must be a valid user of the environments you are connecting to. The test will run under the security roles that the specific user belongs to. You need version 1.200 or newer for this feature to be functional.
Run a test in the context of a specific company

The **General** tab of the Excel parameter file also allows you to specify the name of a legal entity (Company). The test will run in the context of this company. You can specify your default company in the **Settings** dialog box of the tool.

Pause after a specific test step

You can insert a pause between specific test steps. Navigate to the **TestCaseSteps** tab of the Excel parameters file and insert a value (in seconds) in the pause column of a test step. This will pause test cases execution after the test step is completed.

If you don’t see the Pause column, you are using an older version of the Excel parameters file and need to regenerate it. Select the desired test case, then go to **New > Generate Test Execution and Parameter Files**. This may override edits you have made to the parameters file, so you should back up the existing Excel file first.

Other notable test case execution settings

You may find the following settings useful. They are available on the **General** tab of the Excel parameter file.

- **Fail on warning message in the Infolog**: By default, test cases fail when an error occurs or a validation step fails. If you also want a test case to fail in response to a warning message, set the **Fail on warning message in the Infolog** option to **True**. This is useful, for example, if a test case adds a duplicate customer record. The default setting is **False**.
- **Abort test suite execution on failure**: If you set the **Abort test suite execution on failure** option to **True**, execution of the test suite is aborted if the test case fails. All the remaining test cases will have a status of **Not Executed**. The default setting is **False**.
- **Pause between steps**: The number of seconds to pause between test steps. This will affect every test step. The default value is 0 (zero).

Infolog and message validation

Excel parameter files that are generated using version 1.200 or newer contain a **MessageValidation** tab.
You can enter messages in this tab under **Message Validation**. After a test case completes execution, it validates that the messages specified here appear in the Infolog. The test case will fail if these messages are not found.

You can specify any expected messages including error messages. Any message specified in this section will cause a test case to fail unless it is found in the Infolog during execution. Two operators are available: **Equals** and **Contains**. If you use **Equals**, then RSAT performs a string comparison with all messages in the Infolog and fails validation if the full message is not found. If you use **Contains**, then RSAT will validate that at least one message in the Infolog contains the string you specify.

![Message Validation Table](image)

You can configure whether string comparison is case sensitive or not in the **Optional** page of the **Settings** tab.

**Run**

Select **Run** to execute the selected test cases. Only test cases with existing automation files can be run. The tool will open and execute these tests with the data you entered in Excel.

You can modify the order in which test cases are executed using the up and down arrow buttons.

**Pause prior to a test case run**

You can add a pause before a test case starts execution. If you want to pause, update the cell **Pause (seconds)** on the **General** tab of the Excel parameters file of the desired test case.

**Stop a run**

When a test run is in progress, you can select the **Stop** button on the toolbar to cancel the run. Execution stops after the currently running test case completes. The remaining test cases will be marked as **Not Executed** in Azure DevOps.

**Validate readiness of test automation files**

Optionally, you can turn on a setting that validates whether your test cases are ready for execution. This setting prevents unknown errors related to the validity of recordings and test automation files. This option is available as of RSAT version 1.210. You can enable this by selecting the **Settings** tab and then selecting the **Optional** tab.
When enabled, a background process continuously validates the following for each test case.

- The local working directory exists.
- The Excel parameter file exists.
- Test automation files (binary and Xml files) needed for execution exist.
- Test automation files are compatible with current version of RSAT. You must regenerate test automation files when you install a new version of RSAT.
- Test case ID specified in the Excel parameter file matches the test cases ID in Azure DevOps.

The Valid column in the grid indicates the result of the validation process. If validation fails, click on the X in the Valid column to view the error and recommended action.
Investigate results

When all test cases complete execution, **Pass** or **Fail** will be populated in the **Result** column. You can click on the result to see error messages.

Additional investigation details are available in Azure DevOps. To view this information, from your Azure DevOps project page, go to **Test > Runs**.

Select the desired test run. It will include the results of all tests that were executed during that run.
You can open a failed test result and review the **ErrorMessage** section for information about the failure.

### Summary

- **Failed**

- **Run by** not available
- **Test Plan** 28
- **Priority** 0
- **Test suite** Acceptance Test Suite 1
- **Test Case** Business Process Test Demo | Create Purchase Order
- **Configuration** Windows 10

### Error message

**Steps:**
- Navigate to: Purcheditlines (purchformletter_packingslip)
- Click Select.
- In the list, mark the selected row.
- In the Criteria field, type a value.
- Click OK.
- In the list, mark the selected row.
- In the Product receipt field, type a value.

**Warning:** Product receipt PRL was already used as on date 12/7/2018.
- Click OK.
- **Error:** Posting

**Error:** An error occurred during update

**Information:** Operation canceled: Product receipt posting

**ERROR:**
- **Infolog contains:**
- **Error:** Posting
- **Error:** An error occurred during update

All error messages are also available locally under

`C:\Users\$YourUserName\AppData\Roaming\regressionTool\errmsg-<TestCaseId>.txt`

### Test response times

In addition to execution logs, the duration of a test case is also available in the test result.

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Test Case Title</th>
<th>Priority</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passed</td>
<td>(Business Process Test Demo) Create Purchase Order</td>
<td>0</td>
<td>00:1:27.247</td>
</tr>
</tbody>
</table>

You can also review the response time of each step of the test case by opening the **BaseTime.xml** file attached to the test result.

### Attachments (2)

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create_Purchase_Order_40_BaseTime.xml</td>
<td>... 4K</td>
</tr>
<tr>
<td>Create_Purchase_Order_40_BaseLog.txt</td>
<td>... 1K</td>
</tr>
</tbody>
</table>

You need version 1.200 or newer for response times to be available.

### Upload to Azure DevOps to commit your work

To commit your work to Azure DevOps, select **Upload**. This uploads recordings and test automation files, including Excel test parameter files, of all selected test cases to Azure DevOps for future use. After test
automation files are uploaded to Azure DevOps, the next time you use the Regression suite automation tool, even from a different computer, you can simply use **Load** and then **Run**, without generating test execution files or editing Excel parameter files.

In the upload menu, you also have the option to upload recording files (Task recordings) only.

If you are unsure what test cases to select, and you want to commit all changes (since last load) to Azure DevOps, select **Upload all modified automation files** in the upload menu.

### Process compliance

RSAT provides capabilities for managing the readiness of test cases. It also provides a sign-off process for test runs. This is configurable in the **Process** tab under Settings.

**Enforce test case readiness**

You can set up the test case so that it isn’t run unless it has a status of **Ready** in Azure DevOps. Select the **Enforce test case readiness** check box. By default, the check box is cleared.

**Signoffs**

When your test run is complete, RSAT can create sign-off work items in Azure DevOps. Select the **Sign-off tasks** check box. Then set the type of work item that should be created for each person who signs off. You can select the **Functional**, **IT Manager**, or **Team Manager** role for sign-offs, and then specify appropriate email addresses. Work items will then be created in Azure DevOps and assigned to owners for approval.
Regression suite automation tool (RSAT) version 2.2 and later lets you maintain test cases and attachments in the tool itself. In earlier versions, you had to use Microsoft Azure DevOps to maintain test cases and then switch to RSAT to run tests. Therefore, newer versions offer better usability and help improve productivity. Many operations can be done completely in RSAT, and it’s also easier to work with test suites.

Test plans and test suites continue to be maintained in Azure DevOps.

To use this feature, you must enable the Enable upload to Azure DevOps option. Changes that are made in RSAT are then automatically uploaded to Azure DevOps and will be available there. Therefore, test suites will include the updated test cases that are available to other users or that can be run in Azure DevOps by a pipeline.

**View test case information**

Follow these steps to view information about a test case.

1. In the **Test Cases** grid, find the relevant test case, and hover over the row until an ellipsis button (…) appears between the **Title** and **Parameters File** columns.

2. Select the ellipsis button. The menu that appears has two commands: **Open test case** and **Delete Test Case**.
Create a test case that has attachments

3. Select **Open test case** to open the **Test Case information** dialog box.

The **Test Case information** dialog box shows the following information about the test case:

- The name that is assigned to the test case in the test suite appears at the top of the dialog box and can be changed.
- The recording name appears under the test case name. This name is taken from the recording XML file that was used when the recording was made in Task Recorder in the Finance and Operations app or by using the point of sale (POS) client.
- The **Attachments** grid shows the list of attachment files that are available with the test case. You can also find this list by using the **Directory** action under the attachments subfolder.

**Create a test case that has attachments**

Follow these steps to add a new test case by using RSAT.

1. Select the test suite that you want to add a new test case to (**Procure to Pay – v2** in this example). Then select **New Test Case** to open the **Test Case information** dialog box.
2. Enter the name of the test case, and add attachment files. These files include the recording XML file that contains steps for the test case. To add attachment files, select Add, and then, in the dialog box that appears, select the files to add as attachments.

3. When you’ve finished, select Save to save the new test case or Cancel to discard it.

When you save a new test case, RSAT copies the attachment files that you selected into your local RSAT working directory. It maintains the copies there so that they can be used with the test case.

There is no feature that automatically clones test cases from one test suite to another. However, you can manually clone test cases by following these steps.

1. Create a test case as described in the previous procedure. As part of this step, add the recording XML file.
2. Save the new test case, and make a note of the CaseID value that is assigned to it.
3. You can add a parameter Excel file to the new test case. However, the file name must match the new CaseID value. Copy the parameter Excel file from the test case that you’re cloning, and change the file name of the copy so that it matches the new CaseID value.
4. Open the new parameter Excel file, and change all instances of the old CaseID value to the new CaseID value.
5. After you’ve finished updating the new parameter Excel file, add it to the new test case as an attachment.

Alternatively, you can generate a parameter Excel file for the new test case first, and then manually edit it so that it matches the parameter Excel file of the test case that you’re cloning.

**Remove an attachment from a test case**

You can remove attachments from a test case when you no longer require them.
In the **Test Case information** dialog box, select and hold (or right-click) the row for the attachment file, and then select **Remove**.

You can also use this procedure if you’ve edited the recording XML file and you want to upload the new version to the test case. In this case, you should first remove the existing file and then add the new file.

### Delete a test case

Follow these steps to delete a test case.

1. In the **Test Cases** grid, find the relevant test case, and hover over the row until an ellipsis button (...) appears between the **Title** and **Parameters File** columns.

2. Select the ellipsis button, and then select **Delete Test Case** on the menu.

3. Confirm that you want to delete the test case, and optionally specify a reason for the deletion.

A test case that you delete in RSAT is removed from the current test suite, both locally and in Azure DevOps.

In Azure DevOps, work items represent test cases, and test suites contain links to the test case work items. A test case is reused by linking to it from more than one test suite. When a test case is deleted in RSAT, RSAT determines whether the test case is linked to one test suite or more than one test suite. If the test case is used only by the current test suite, RSAT deletes the Azure DevOps work item that represents the test case. If the test case is used by other test suites, RSAT doesn’t delete the work item itself. Instead, it deletes only the link to the work item.
An important component of a test case is validation of expected values. You can define validation parameters during the authoring of your test cases using Task Recorder. While recording, right-click on a control and select **CurrentValue** under the **Task Recorder > Validate** menu. This action becomes a validation step that you can use with the Regression suite automation tool. The control value will become a validation variable in the automatically generated Excel parameters file. The menu item is shown in the following image.

For more information about how to create task recordings, see [Task recorder resources](#).

When RSAT generates the Excel parameter file for a test case, validation steps are added as shown in the image below. You can enter the expected value to use during execution of the test case.

![Excel parameter file with validation steps](#)

**Validate expected values using operators**

You can also use operators in validation steps to validate that a variable is not equal, less than, or greater than a specified value. To use this feature, open the **Settings** tab and select the **Optional** tab. Turn on the setting named **Use operators for validation**. This option is available as of RSAT version 1.210. If you have been using
an older version of the tool, you must regenerate new Excel parameter files to take advantage of this functionality. In the Excel file, a new **Operator** field will appear, as shown in the following image.

When recording test cases, Task Recorder supports additional validation action:

- Validate whether a control is enabled or disabled.
- Validate whether a control is editable or read-only.

To take advantage of this validation, you need to be use a Finance and Operations app running on 10.0.13 (or newer) and RSAT 2.0 (or newer). For more information, see Validate.
One of the key features of the Regression Suite Automation Tool is the chaining of test cases, that is, the ability of a test to pass values to other tests. Test cases are executed according to their defined order in the Azure DevOps test plan, which can also be updated in the test tool itself. It is important to correctly order the tests if you want to pass variables from one test case to the other.

To save the value of a variable while recording the test in Task Recorder, right-click the field and select Task recorder > Copy, as shown in the following image. Copying will save the variable in the recording file. This variable can be used in subsequent tests.

When RSAT generates the Excel parameters file, saved variables appear in the Saved variables table on the General Tab. These variables also appear in the context of the test case steps in the TestCaseSteps tab. In the image below, the purchase order ID value was copied during the recording of the test case (step 5). This value is stored in a variable named ({{PurchCreateOrder_PurchTable_PurchId_86_Copy}}).
Support for formulas of saved variables

To reuse these variables during test playback, copy the variable name and use it in place of a parameter value in the data file of another test (or the same test), as shown below.

Variables can be used in the same test case where they are defined and can also be passed between tests during the same test run.

Support for formulas of saved variables

You can create formulas that contain saved (copied) variables. If you have been using an older version of the Regression Suite Automation Tool, you will need to regenerate new Excel parameter files to take advantage of this functionality. Supported operators are +, -, /, and ''. Only numerical variables can be used in the Regression Suite Automation Tool formulas. Strings or dates are not supported. Always specify variable names within double braces {{varname}}. For example, {{var1}} + {{var2}}.

In the image below, two different variables are used in a formula.
As of RSAT version 1.220, you can also use Excel functions, such as ROUND, CONCAT, and UPPER, to create formulas with RSAT variables. This feature is implemented using the Excel formula evaluation functionality, so any function supported by Excel is supported by RSAT.

For example,

- To round a value into the nearest whole number, use:
  
  ```excel
  =ROUND({{Item_Price_3274_Copy}}, 0)
  ```

- To concatenate strings, use:
  
  ```excel
  =CONCATENATE({{AccountNum_3274_Copy}}, " ", {{ AddressBP_Locator_3274_Copy}})
  ```

- To calculate and format a date and convert it to a string, use:
  
  ```excel
  =TEXT(DATEVALUE({{SystemDate_CurrentDate_3276_Copy}}) - 1, "mm/dd/yyyy")
  ```

Always convert RSAT date values to text for reliable test case execution.

RSAT evaluates these formulas during test execution, so you must precede the formula with a single quote `’` to prevent Excel from attempting to prematurely calculate the formula. An example is shown in this image.

Use variables in message validation

You can also use a saved variable as part of a string in the Message Validation tab. Here is an example that validates that the message Customer account {{variable name}} already exists. It appears in the Infolog during test execution. {{variable name}} is a variable that is copied during the recording.

Saved (Copied) variables can be used within the same test case or across more than one test case in the same test suite.
<table>
<thead>
<tr>
<th></th>
<th>Message Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Messages</td>
</tr>
<tr>
<td>3</td>
<td>Customer account ([var name]) already exists.</td>
</tr>
</tbody>
</table>
The Regression suite automation tool (RSAT) lets you use the same task recording with multiple test cases, so that you can run a task with different data configurations. Select a test case in the Regression suite automation tool and then select New > Create Derived Test Case. This creates a child test case in Azure DevOps. The resulting derived test case is linked to its parent test case in Azure DevOps. It has an Excel parameters file attached but no recording file. The derived test case will appear in the Regression suite automation tool grid under the same test suite with the Derived column selected. By default, derived test cases are named after their parent test case with a numeric suffix.

In the following image, a derived test case has been created from a test case named Create a Sales Order and Validate - v2. The derived test case has been renamed (in Azure DevOps) to Create a Sales Order and Validate - v2 (Fail validation).

In Azure DevOps, a derived test case is a child item of the Create a Sales Order and Validate - v2 test case and is tagged with the special keyword RSAT:DerivedTestSteps.

When you run a derived test case, it will use the recording of its parent test case and its own copy of the Excel parameters file. This will allow you to run the same test with different parameters without the need to maintain more than one recording.
A derived test case does not need to be part of the same test suite as its parent test case, and you can use it in another suite. You can also rename a derived test case. You can edit the Excel parameters file of a derived test case to run it with a different user, a different company, or with different input and validation parameters than its parent test case.
The format of parameter Microsoft Excel files that are used with Regression suite automation tool (RSAT) changed in the 2.0 release. The format is now more intuitive and shows test steps. New test cases that are created in RSAT version 2.0 and later automatically generate parameter files in the new format. However, you might have tests that were created before version 2.0 and that still use the old format. RSAT will continue to support running test cases that use parameter files in the old format at least until the next major release. RSAT can also upgrade old parameter files to the new format.

Parameter files can be freely edited in Excel. In some situations, a parameter file can't be fully upgraded, and some manual work is required to complete the process. The next section explains the mechanics of the upgrade process.

**Upgrade process**

The upgrade process tries to do a full upgrade. However, in some situations, this process can't be safely completed.

When the full upgrade process can be completed, the old parameter file is replaced with a new one. A copy of the original file is created, and _BAK is appended to the file name.

If only part of the parameter file can be safely upgraded, the original file isn't changed. Instead, a new file is created, and _PARTIAL is appended to the file name. The partial file contains the information that could be upgraded. You can use the partial file to manually transfer missing parts from the original file and complete the new file. When you've finished, rename the original file to a backup name (for example, append _BAK to the file name). Then rename the new file by removing _PARTIAL from the file name. The new file then becomes the new parameter file.

You can continue to use the old parameter file until the new file is ready. RSAT will continue to support the old format for parameter files at least until the next major release.

The upgrade process can usually upgrade unedited parameter files. It can also upgrade files where only cell values have been changed in the file. However, if additional cells have been added and referenced, the upgrade process can't be completed automatically. In this case, the partial file will include cells that have the value #MISSING. This value indicates that cell references are missing. You must manually add the information from the original parameter file to the new partial file.

From now on, add new cells to the new **CustomParameters** sheet in the parameter file, just as they are added in the partial file.

In RSAT release 2.2 and later, the parameter file has a sheet that is named **CustomParameters**. This sheet is included to help future-proof the upgrade of parameter files. If you add cells, add them to this sheet.

If you added a cell to the old parameter file and assigned a name to that cell, the named cell is automatically moved to the **CustomParameters** sheet during the upgrade, provided that its value doesn't reference other
After you’ve finished moving your added cells to the partial file, if any references on the **TestCaseSteps** sheet have #MISSING as the value, change the references so that they match the relevant cells on the **CustomParameters** sheet. Ideally, you should always reference cells by their assigned name (for example, **MyQuantity**), not by the cell identifier (for example, **E4**).

NOTE

As a best practice, you should assign names to any cells that are added to the **CustomerParameters** sheet. Then reference the cells by name from test case steps.

After the upgrade, you should run your test cases to make sure that the new parameter file produces the expected results. Complete this step both when the upgrade was fully completed and when the upgrade required manual work.

When you’ve finished creating and testing the new files, you can delete the backup and partial files.

**Run the parameter file upgrade process**

To upgrade the parameter files, follow these steps.

1. Open RSAT.
2. Select that test cases that have the parameter files that you want to upgrade.
3. On the **New** menu, select **Upgrade Parameter files (will auto generate Test Execution files)**.

The parameter files for all the selected cases are upgraded.

The upgrade process skips test cases where the parameter file is already in the new format. A parameter file is considered upgraded if it contains the **CustomParameters** sheet.

When the upgrade process is completed, a message box appears that shows a summary. The summary includes the following information:

- The total number of selected test cases that were marked for upgrade.
- The number of successful upgrades. For these upgrades, there are new parameter files and _BAK files.
- The number of failed upgrades. For these upgrades, there are new _PARTIAL files.
- The number of skipped upgrades.
An explanation of the results.

The following illustration shows an example of the summary message box.

For a failed upgrade, you can find more information by selecting the yellow triangular warning symbol next to the test case title. The following illustration shows an example of the message box that appears.

IMPORTANT

You can run the upgrade repeatedly. In this case, newly upgraded parameter files will be skipped. However, new partial files will overwrite existing partial files. We recommend that you complete all partial files and rename them before you rerun the upgrade.
This tutorial walks through some of the advanced features of the Regression suite automation tool (RSAT), includes a demo assignment, and describes strategy and key learning points.

**Notable Features of RSAT and Task recorder**

**Validate a field value**

RSAT allows you to include validation steps within your test case to validate expected values. For information about this feature, see the article [Validate expected values](#).

The following example shows how you can use this feature to validate whether the on-hand inventory is more than 0 (zero).

1. In the demo data in the **USMF** company, create a task recording that has the following steps:
   a. Go to **Product information management > Products > Released products**.
   b. Use the Quick Filter to find records. For example, filter on a value of **1000** for the **Item number** field.
   c. Select **On-hand inventory**.
   d. Use the Quick Filter to find records. For example, filter on a value of **1** for the **Site** field.
   e. In the list, mark the selected row.
   f. Validate that the value of the **Total available** field is **411.0000000000000000**.

2. Save the task recording as a **developer recording** and attach it to your test case in Azure Devops.

3. Add the test case to the test plan, and load the test case into RSAT.

4. Open the Excel parameter file and go to the **TestCaseSteps** tab.

5. To validate whether the inventory on-hand will always be more than 0, go to the **Validate Total Available** step and change its value from **411** to **0**. Change the value of the **Operator** field from an **equal sign (=)** to a greater than sign (**>**).


7. Select **Upload** to save the changes that you made to the Excel parameter file to Azure DevOps.

Now, if the value of the **Total Available** field for the specified item in inventory is more than 0 (zero), tests will pass, regardless of the actual on-hand inventory value.

**Saved variables and chaining of test cases**

One of the key features of RSAT is the chaining of test cases, that is, the ability of a test to pass variables to other tests. For more information, see the article [Copy variables to chain test cases](#).

**Derived test case**

RSAT lets you use the same task recording with multiple test cases, enabling a task to run with different data configurations. See the article [Derived test cases](#) for more information.
**Validate notifications and messages**

This feature can be used to validate whether an action occurred. For example, when a production order is created, estimated, and then started, the app shows a "Production – Start" message to notify you that the production order has been started.

You can validate this message through RSAT by entering the message text on the `MessageValidation` tab of the Excel parameter file for the appropriate recording.

After the test case is run, the message in the Excel parameter file is compared to the message that is shown. If the messages don't match, the test case will fail.

**NOTE**

You can enter more than one message on the `MessageValidation` tab in the Excel parameter file. The messages also can be error or warning messages instead of informational messages.

**Snapshot**

This feature takes screenshots of the steps that were performed during task recording. It is useful for auditing or debugging purposes.

- To use this feature while running RSAT with the user interface, open the `Microsoft.Dynamics.RegressionSuite.WindowsApp.exe.config` file under the RSAT installation folder (for example, `C:\Program Files (x86)\Regression Suite Automation Tool`), and change the value of the following element from `false` to `true`.

  ```xml
  <add key="VerboseSnapshotsEnabled" value="false" />
  ```

- To use this feature while running RSAT by the CLI (for example, Azure DevOps), open the `Microsoft.Dynamics.RegressionSuite.ConsoleApp.exe.config` file under the RSAT installation folder (for example, `C:\Program Files (x86)\Regression Suite Automation Tool`), and change the value of the following element from `false` to `true`.

  ```xml
  <add key="VerboseSnapshotsEnabled" value="false" />
  ```

When you run test cases, RSAT generates snapshots (images) of the steps and saves them in the playback folder of the test cases in the working directory. In the playback folder, a separate subfolder is created named `StepSnapshots`. That folder contains snapshots for the test cases that are run.
Assignment

Scenario
1. The product designer creates a new released product.
2. The production manager initiates a production order to bring the stock level to two pieces.
3. Manufacturing starts and ends the production order, and verifies that the on-hand quantity is two pieces.
4. The sales team receives an order for four pieces of the new product. Therefore, the sales team updates the net requirements via the dynamic plan. Because no additional capacity is available, the default order policy is set to “buy instead of make.” Therefore, a planned purchase order is created.
5. The buyer adds a vendor, firms the planned purchase order, and then confirms the purchase order.
6. When the goods that were purchased arrive at the store, the store operator searches the related purchase order and receives the goods. Because the order is now completed, goods can be picked and packed against the sales order.
7. Finance posts the purchase invoice and sales invoice.

The following illustration shows the flow for this scenario.

The following illustration shows the business processes hierarchy for this scenario in the LCS Business Process Modeler.
**Strategy – Key learning**

**Data**

- Make sure that you have representative data volumes (a copy of production/golden configuration data plus migrated data).

- When you generate new data via Task recorder, create test names that won’t conflict with existing names (for example, use a prefix such as RSATxxx).

- Use Azure Point-In-Time restore to rerun tests in non-Tier 1 environments.

- Although you can use the **RANDOM** and **NOW** Excel functions to generate a unique combination, the effort is considerably high. Here is an example.

    ```
    product =  "AT" &TEXT(NOW(),"yyymmdhhmm")
    ```

**Task recorder**

- Define scenarios before you start recording. A well-managed project has predefined test scenarios. To build a test case, consider how predictable the outcome of those test scenarios is.

- Split recordings if they are performed by different roles, or if there is waiting time or an external event before the next step.

- Avoid selecting values in lists. Instead, use text formats, such as **FIFO**, **AudioRM**, and **SiteWH**. When you select in a list, the position of the value in the list is recorded, not the value itself. If items are added to that list, the position of the value can change. Therefore, your recording will use a different parameter, and the rest of the scenario might be affected.

- Think about multi-user behavior. For example, don’t assume that your newly created sales order will always be automatically selected. Instead, always use the filter to find the correct order.

- Use the Copy function in Task recorder to save the name of a newly created product so it can be used in...
chained test cases.

- Use the Validate function in Task recorder to set checkpoints that verify that steps have been run correctly.

**RSAT**

- To run the test in another company, you can change the company on the *General* tab of the Excel parameter file. Make sure that settings and data are available in the newly selected company.
- You can change the test user on the *General* tab of the Excel parameter file. Specify the email ID of the user who will run the test case. In this way, the test case can be run by using the security permissions of the specified user.
- To wait before the test is started, you can define a pause on the *General* tab of the Excel parameter file. This pause can be used in a batch job (for example, if a workflow must be run before the next step can be performed.)

**Advanced scripting**

**CLI**

RSAT can be called from a *Command Prompt* or *PowerShell* window.

```
NOTE
Verify that the TestRoot environment variable is set to the RSAT installation path. (In Microsoft Windows, open Control Panel, select System and Security > System > Advanced system settings, and then select Environment Variables.)
```

1. Open a *Command Prompt* or *PowerShell* window as an admin.

2. Navigate to the RSAT installation directory.

   ```
   cd "c:\Program Files (x86)\Regression Suite Automation Tool"
   ```

3. List all commands.
C:\Program Files (x86)\Regression Suite Automation

Usage:
   Microsoft.Dynamics.RegressionSuite.ConsoleApp.exe command
or
   Microsoft.Dynamics.RegressionSuite.ConsoleApp.exe /settings "C:\Path to\file.settings" command

Available commands:
   ?
   about
   cls
   download
   edit
   generate
generatederived
generatetestonly
generatetestsuite
   help
   list
   listtestplans
   listtestsuite
   listtestsuitenames
   playback
   playbackbyid
   playbackmany
   playbacksuite
   quit
   upload
   uploadrecording
   usage

?  Shows help about all available commands and their parameters.


? : Optional parameters

command : Where [command] is one of the commands specified below.

about
   Displays the current version.

Microsoft.Dynamics.RegressionSuite.ConsoleApp  about

cls
   Clears the screen.

Microsoft.Dynamics.RegressionSuite.ConsoleApp  cls

download
   Downloads attachments for the specified test case to the output directory. You can use the list command to get all available test cases. Use any value from the first column as a test_case_id parameter.

Microsoft.Dynamics.RegressionSuite.ConsoleApp  download  [test_case_id] [output_dir]
download: required parameters

   test_case_id : Represents the test case ID.
   output_dir  : Represents the output directory. The directory must exist.

download: examples

download 123 c:\temp\rsat

download 765 c:\rsat\last
edit

Allows you to open parameters file in Excel program and edit it.

Microsoft.Dynamics.RegressionSuite.ConsoleApp edit [excel_file]

eedit: required parameters
  • excel_file: Must contain a full path to an existing Excel file.

eedit: examples
  edit c:\RSAT\TestCase_123_Base.xlsx
  edit e:\temp\TestCase_456_Base.xlsx

generate

Generates test execution and parameter files for the specified test case in the output directory. You can use the `list` command to get all available test cases. Use any value from the first column as a `test_case_id` parameter.

Microsoft.Dynamics.RegressionSuite.ConsoleApp generate [test_case_id] [output_dir]

generate: required parameters
  • test_case_id: Represents the test case ID.
  • output_dir: Represents the output directory. The directory must exist.

generate: examples
  generate 123 c:\temp\rsat
  generate 765 c:\rsat\last

generatederived

Generates a new test case, derived from the provided test case. You can use the `list` command to get all available test cases. Use any value from the first column as a `test_case_id` parameter.

Microsoft.Dynamics.RegressionSuite.ConsoleApp generatederived [parent_test_case_id] [test_plan_id] [test_suite_id]

generatederived: required parameters
  • parent_test_case_id: Represents the parent test case ID.
  • test_plan_id: Represents the test plan ID.
  • test_suite_id: Represents the test suite ID.

generatederived: examples
  generatederived 123 8901 678

generatetestonly

Generates only test execution file for the specified test case in the output directory. You can use the `list` command to get all available test cases. Use any value from the first column as a `test_case_id` parameter.

Microsoft.Dynamics.RegressionSuite.ConsoleApp generatetestonly [test_case_id] [output_dir]

generatetestonly: required parameters
  • test_case_id: Represents the test case ID.
  • output_dir: Represents the output directory. The directory must exist.

generatetestonly: examples
  generatetestonly 123 c:\temp\rsat
  generatetestonly 765 c:\rsat\last

generatetestsuite

Generates all test cases for the specified suite in the output directory. You can use the `listtestsuitenames` command to get all available test suits. Use any value from the column as a `test_suite_name` parameter.
`generatetestsuite`: required parameters

- `test_suite_name`: Represents the test suite name.
- `output_dir`: Represents the output directory. The directory must exist.

`generatetestsuite`: examples

```
generatetestsuite Tests c:\temp\rsat

generatetestsuite Purchase c:\rsat\last
```

`help`

Identical to the `?` command.

`list`

Lists all available test cases.

```
list
```

`listtestplans`

Lists all available test plans.

```
listtestplans
```

`listtestsuite`

Lists test cases for the specified test suite. You can use `listtestsuitenames` command to get all available test suites. Use any value from first column as `suite_name` parameter.

```
listtestsuite [suite_name]
```

`listtestsuitenames`

Lists all available test suites.

```
listtestsuitenames
```

`playback`

Plays back a test case using an Excel file.

```
playback [excel_file]
```

`playbackbyid`

Plays back multiple test cases at once. You can use the `list` command to get all available test cases. Use any value from the first column as a `test_case_id` parameter.

```
playbackbyid [test_case_id1] [test_case_id2] ... [test_case_idN]
```
playbackbyid: examples

playbackbyid 878

playbackbyid 2345 667 135

**playbackmany**

Plays back many test cases at once, using Excel files.

```
Microsoft.Dynamics.RegressionSuite.ConsoleApp playbackmany [excel_file1] [excel_file2] ... [excel_fileN]
```

**playbackmany: required parameters**

- **excel_file1**: Full path to the Excel file. File must exist.
- **excel_file2**: Full path to the Excel file. File must exist.
- **excel_fileN**: Full path to the Excel file. File must exist.

**playbackmany: examples**

```
playbackmany c:\RSAT\TestCaseParameters\param1.xlsx
playbackmany e:\temp\test.xlsx f:\rsat\sample1.xlsx c:\RSAT\sample2.xlsx
```

**playbacksuite**

Plays back all test cases from the specified test suite. You can use `listtestsuitenames` command to get all available test suites. Use any value from first column as `suite_name` parameter.

```
Microsoft.Dynamics.RegressionSuite.ConsoleApp playbacksuite [suite_name]
```

**playbacksuite: required parameters**

- **suite_name**: Name of the desired suite.

**playbacksuite: examples**

```
playbacksuite suiteName
playbacksuite sample_suite
```

**quit**

Closes the application.

```
Microsoft.Dynamics.RegressionSuite.ConsoleApp quit
```

**upload**

Uploads all files belonging to the specified test suite or test cases.

```
Microsoft.Dynamics.RegressionSuite.ConsoleApp upload [suite_name] [testcase_id]
```

**upload: required parameters**

- **suite_name**: All files belonging to the specified test suite will be uploaded.
- **testcase_id**: All files belonging to the specified test case(s) will be uploaded.

**upload: examples**

```
upload sample_suite
upload 123
upload 123 456
```

**uploadrecording**

Uploads only recording file belonging to the specified test cases.

```
```
uploadrecording: required parameters

- **testcase_id**: Recording file belonging to the specified test cases will be uploaded.

uploadrecording: examples

- `uploadrecording 123`
- `uploadrecording 123 456`

usage

Shows two ways to invoke this application: one using a default setting file, another one providing a setting file.

`Microsoft.Dynamics.RegressionSuite.ConsoleApp | usage`

**Windows PowerShell examples**

**Run a test case in a loop**

You have a test script that creates a new customer. Via scripting, this test case can be run in a loop by randomizing the following data before each iteration is run:

- Customer ID
- Customer name
- Customer address

The customer ID will be in the format *ATCUS<number>*\*, where `<number>` is a value between *00000001* and *999999999*.

The following example uses one parameter, `start`, to define the first number that is used. Is uses a second parameter, `nr`, to define the number of customers that must be created. For each iteration, the parameters in the Excel parameter file are changed by using an `UpdateCustomer` function. Then the RSAT command line is called in a `RunTestCase` function.

Open Microsoft Windows PowerShell Integrated Scripting Environment (ISE) in admin mode, and paste the following code into the window that is named `Untitled1.ps1`.
param ( [int]$start = 1, [int]$nr = 1 )
function UpdateCustomer
{
    param ([string]$paramFilename, [string]$sheetName, [string]$CustId)
    $xl = New-Object -COM "Excel.Application"
    $xl.Visible = $false
    $wb = $xl.Workbooks.Open($paramFilename)
    $ws = $wb.Sheets.Item($sheetName)
    $ws.Cells.Item(3, 2).Value = "ATCUS" + $CustId
    $ws.Cells.Item(4, 2).Value = "Automated Test Customer " + $CustId
    $ws.Cells.Item(8, 2).Value = "Automated Test Street " + $CustId
    $wb.Save()
    $wb.Close()
    $xl.Quit()
    [System.Runtime.Interopservices.Marshal]::ReleaseComObject($xl)
}
function RunTestCase
{
    param ( [string]$filename )
    $cmd = "cd c:\Program Files (x86)\Regression Suite Automation Tool\ &&  "
    $cmd = $cmd + "Microsoft.Dynamics.RegressionSuite.ConsoleApp.exe playback "
    $cmd = $cmd + $filename
    cmd /c $cmd
}
$excelFilename = "full path to Excel parameter file"
$sheetName = "DirPartyQuickCreateForm"
for ($i = $start; $i -lt $start + $nr; $i++ )
{
    $CustomerId = $i.ToString("000000000")
    Write-Host "customer : " $CustomerId
    UpdateCustomer $excelFilename $sheetName $CustomerId
    RunTestCase $excelFilename

Run a script that depends on data in Microsoft Dynamics 365
The following example uses an Open Data Protocol (OData) call to find the order status of a purchase order. If
the status isn't invoiced, you can, for example, call an RSAT test case that posts the invoice.
function Odata_Get {
    Param ( [string] $environment, [string] $cmd )
    $tenant = "your tenant"
    $creds = @{
        grant_type = "client_credentials"
        client_id = "your client application Id"
        client_secret = "your client secret"
        resource = $environment
    }
    $headers = $null
    $headers = @{
        Authorization = "Bearer " + $bearer.access_token
    }
    $Odata_cmd = $environment + '/data/' + $cmd
    return (Invoke-RestMethod -Uri $Odata_cmd -Method Get -Headers $headers -ContentType application/json )
}
function PurchaseOrderStatus {
    Param ( [string] $environment, [string] $purchaseOrderNumber )
    $cmd = 'PurchaseOrderHeaders?$filter=PurchaseOrderNumber eq '
    $cmd = $cmd + $'" + $purchaseOrderNumber + "'
    $response = Odata_Get -environment $environment -cmd $cmd
    return $response.value.PurchaseOrderStatus
}
$environment = "https://your environment"
$orderStatus = PurchaseOrderStatus -environment $environment -purchaseOrderNumber '000003'
if ($orderStatus -eq $null) { write-host 'doesn't exist'}
elseif ($orderStatus -ne 'invoiced') { RunTestcase "PostInvoice" }
The Regression suite automation tool (RSAT) uses privileged resources on the machine that it is running on. A user must be an administrator on the machine to run RSAT tests. This topic explains how to grant these privileged resources to users if you are using **RSAT version 2.2 or later**. The non-administrator user can run RSAT tests without being an administrator on the machine.

These instructions will not allow a non-administrator user to install RSAT. The instructions only enable using RSAT after it has been installed. This situation includes first-time use of RSAT where the Selenium framework is installed, or with new browser driver installation after updating browser versions. Those installation steps still require running RSAT with administrator privileges.

When RSAT is installed on a virtual machine (VM) that is shared by multiple users, then users can become blocked when multiple users run RSAT at the same time. For example, a user might hold resources while running tests cases, which then blocks access to other users. These instructions do not change that behavior.

### Enable non-administrator RSAT use

To enable non-administrator RSAT use, you need two PowerShell scripts and a new shortcut file. These files are in the RSAT installation folder in the subfolder named **Enable non admin**.

1. Open Windows PowerShell as Administrator.

2. Change the folder to **Enable non admin** in the RSAT installation folder. The installation folder is named according to the localized Windows running on the machine, for example `C:\Program Files (x86)\Regression Suite Automation Tool\Enable non admin`.

3. In the folder **Enable non admin** you will find these files:
4. The first file, **Enable-non-admin-mode.ps1** is a PowerShell script that enables non-administrator mode for the machine. Run this file once on each machine.

Run the script **Enable-non-admin-mode.ps1** directly from the *Enable non admin* folder, using these required parameters:

- **action**: (string) Option to enable or disable non-administrator mode. Valid values are *enable* and *disable*.
- **thumbprint**: (string) Certificate thumbprint. This value must be the same value that is specified in RSAT under General settings.

Here's an example:

```
.\Enable-non-admin-mode.ps1 "enable" "2305555DXXXXXXXXXXXXXXXXXXXXXX"
```

To get help, execute this command.

```
help .\Enable-non-admin-mode.ps1 -full
```

**IMPORTANT**

DO NOT remove or copy this PowerShell script from the *Enable non admin* folder. Run it only from this folder.

5. The second file, **Enable-non-admin-user.ps1** is a PowerShell script that enables non-administrator mode for a user. Run this script for each user that is going to use RSAT on the machine.

Run the script **Enable-non-admin-user.ps1** directly from the *Enable non admin* folder, using these required parameters:

- **action**: (string) Option to enable or disable non-administrator mode. Valid values are *enable* and *disable*.
- **thumbprint**: (string) Certificate thumbprint. This value must be the same value that is specified in RSAT under General settings.
- **user**: (string) The local username in the format `domain\userName`.

Here's an example:

```
.\Enable-non-admin-user.ps1 "enable" "TESTDOMAIN\testuser" "2305555DXXXXXXXXXXXXXXXXXXXXXX"
```

To get help, execute this command.

```
help .\Enable-non-admin-user.ps1 -full
```

**IMPORTANT**

DO NOT remove or copy this PowerShell script from the *Enable non admin* folder. Run it only from this folder.

7. Copy the shortcut file Regression Suite Automation Tool (Non admin).lnk to the user’s desktop. (The old shortcut calls a Visual Basic script that some users may not be allowed to execute. The new shortcut will call the executable file directly.)

**IMPORTANT**

DO NOT remove the old shortcut, because it is a shared file that is used by all users on the machine. If you remove the file, then it will disappear for all users. It is fine to remove the old shortcut if all users will be enabled to run as a non-administrator. However, the shortcut will reappear every time a new version of RSAT is installed.

8. When you start RSAT the first time, you must download and install the Selenium framework and the drivers that match the version of your browser. Users that are not administrators might not be able to download and install these components. In this case, RSAT might fail and generate exceptions. To download and install the components, run RSAT with administrator privileges to complete the installation. To run with administrator privileges, right-click the new shortcut Regression Suite Automation Tool (Non admin) and select Run as administrator. After the installation finishes, close RSAT and then have the user start RSAT again.

9. Use the new shortcut Regression Suite Automation Tool (Non admin) to start RSAT.

**Disable non-administrator RSAT use**

If you need to revert the machine back to run with administrator use, then run the PowerShell script Enable-non-admin-mode.ps1 with the action parameter set to disable.

```
.\Enable-non-admin-mode.ps1 "disable"
```

**Future versions of RSAT**

Currently, we are collecting feedback from users running RSAT with this non-administrator mode. In the future, we might change the installation process to automatically include the steps that enable non-administrator users.
This topic describes best practices and common use cases of the Regression suite automation tool (RSAT) and Task recorder.

Author test cases using the Task recorder

When you author task recordings for RSAT, follow these practices:

1. Make sure all your recordings start on the main dashboard.
2. Keep individual recordings short and focus on a business task performed by one user, like creating a sales order. This simplifies maintainability and reusability of test cases.
3. Chart controls are not supported. Any task recording actions related to charts will be ignored by RSAT during test case playback.
4. When creating a recording, make sure to select a tab header even if the tab is already open. For example, you can switch to another tab and then select the needed tab again to activate it before using a control on it. This will make your recording more reliable during test case playback.
5. RSAT cannot play back any test step that is not recognized by the task recorder. For example, you cannot upload a file from the local disk during play back of a test case.
6. RSAT cannot play back a page refresh step. Avoid refreshing a page while recording your test.

Best practices when using the Regression suite automation tool

1. Upon opening the tool for the first time, select Settings and ensure that you have all the needed settings.
2. Before installing a new version of the tool, it is recommended to close and uninstall the previous version.
3. When you install a new version of the tool, regenerate all test execution files.

It is not necessary to regenerate Microsoft Excel parameter files unless you want to take advantage of new features available in a newer format of parameter files.

4. For test parameters that need a unique value, for example, the product receipt number in the Product Receipt form or the invoice number in the Vendor Invoice form, use the RandBetween(a,b) Excel function to generate a unique number every time the test case is executed.

5. The default values in Excel come from the task recording. For Reference Group controls such as storage dimensions or tracking dimensions, it stores the key of the lookup instead of the value, for example, 2 instead of SiteWH. We recommend that you update these fields with the actual value in Excel so that the test is more robust and resilient to changes.
6. It is recommended to set the same locale for **Language** and **Date, time, and number format** settings of your environment prior to running RSAT. If these values are inconsistent, it may result in validation errors.

RSAT relies on Azure DevOps to store and manage test recording files (also known as task recordings). When RSAT loads a test plan from Azure DevOps, associated files are downloaded to the current **working directory** on your local computer. (This working directory is defined in RSAT settings.)

In version 1.200.42264.6 and later, it's easier to manage local recording files. You can make changes in Task recorder and then use RSAT to test them, without having to go through the Business process modeler (BPM) or Azure DevOps. When you use Task recorder, after you’ve finished authoring or modifying a recording, you can save it directly to your local disk as a developer recording.

Put the recording file under the working directory that is associated with the test case. For example, if your configured working directory is `C:\Users\<username>\Documents\RSAT`, put the recording file for test case 1234 under `C:\Users\<username>\Documents\RSAT\1234\attachments`. You must name the developer recording file **Recording.xml**. Alternatively, you can name the recording file `-Test Case Title-.xml`, where `-Test Case Title-` is the title of the test case in Azure DevOps.

The following illustration shows an example of a working directory folder structure. You can open the directory directly from RSAT by clicking the folder symbol.
Commit a recording file to Azure DevOps

Each test case has its own folder, which is named after the ID of the test case. Test case attachments (recording files, automation files, and Excel parameter files) are downloaded into an attachments folder. Here is an example.

```
P: Documents > 1396 > attachments
```

- Create_Sales_Order_1396_Base.cs
- Create_Sales_Order_1396_Base.dll
- Create_Sales_Order_1396_Base.xlsx
- Create_Sales_Order_1396_Base.xml
- Recording.xml

The generatorLogs directory contains log files. It doesn’t contain any files that users can modify. You can ignore this directory unless RSAT support explicitly asks you to provide log files from it.

**Commit a recording file to Azure DevOps**

After a recording has been tested and finalized, use RSAT to upload it and commit it to Azure DevOps. The upload button has two options: *Upload automation files* and *Upload recording file*. The second option uploads only your recording file to Azure DevOps.

**NOTE**

If you’re using a version of RSAT that is earlier than 1.200.37255.0, and you upgrade to the latest version, you must reload your test cases from Azure DevOps to download them into the correct directory. Otherwise, RSAT will fail, and you will receive a "File not found" error.

If you’re working across several DevOps projects, we recommend that you use a different working directory for each project. Otherwise, attachment files from multiple projects can become commingled in the same directory structure.

Modify (Edit) a Task recording

If you want to modify an existing task recording, note these best practices.

In the web client, open the Task recorder pane and start editing the recording using the *Edit Recording* option.
When you've finished editing the recording, play it back in the client, and verify that all the steps work correctly. Playback is required.

After you've finished playing back an edited recording, save it. It's then ready to be used by RSAT.

Copy test cases in Azure DevOps

As you are building your test suites in Azure DevOps, it is handy and common to duplicate test cases along with their attachments. If a copied test case contains an existing Excel parameter file attached, RSAT cannot execute it without manual edits to the Excel file. The **Test Case ID** in the Excel parameter file must match the Azure DevOps test case ID. You will need to edit all copied Excel parameter files. In the following image, the Excel file is associated with Test Case number 53 in Azure DevOps.

As of RSAT version 1.210, this process is easier. To automatically fix all occurrences of a mismatch, select the desired test cases in the grid, and then select **Resolve test case ID mismatch** in the **New** menu.
While the functional validation of an ERP application can’t be fully data agnostic, there are multiple phases and approaches for testing. These testing phases include:

- **SysTest framework**
  - The SysTest framework is reliable for writing unit tests. Because unit tests are generally testing a method or function, they should always be data agnostic and dependent only on the input data that is provided as part of the test.

- **ATL framework**
  - Microsoft has an ATL framework that is an abstraction on the SysTest framework and makes functional test writing much more simple and reliable. This framework should be used for writing component tests or simple integration tests.

- **Regression Suite Automation Tool (RSAT)**
  - The RSAT is used for integration tests and business cycle tests. The business cycle tests, also called the regression validation tests, are dependent on existing data. However, these tests can become data agnostic if you consider additional factors.

  - Where unit tests and component tests are low level and can fully be data agnostic (not dependent on existing dataset), the business cycle or regression validation tests are dependent on some existing data. This data includes setup, configuration settings (parameters), and master data (customer, vendors, items, etc.), but never transaction data. Make sure that during the test, if any of these are being changed, that they are reverted back as part of the final test.
  - Select master data based on certain criteria instead of selecting a particular record. For example, if you
want to select an item based on its dimension values and stock availability, filter the product list with those values, select the first item, and copy the number to be used for future tests. If it's a simple master data line such as customer, vendor, or item, it can be created as part of the automation and used in future tests through chaining.

- Enter the unique identifiers, such as invoice numbers, through the number sequence or by using Microsoft Excel functions such as =TEXT(NOW(),"yyyymmddhhmm"). This function will provide a unique number every minute, which allows you to track when the action happened. This can be used for variables such as product receipt numbers and vendor invoice numbers. These tests continue to work on the same database again and again, without requiring any restoration.

- Always set the **Edit mode** of the environment to **Read** or **Edit** as the first test case because the **default option is Auto**. The **Auto** options always use the previous setting and can cause unreliable tests.

- Only validate after you filter on a particular transaction instead of generic validation. For example, for the number of records, filter for the transaction number or the transaction date so that the validation excludes all other transactions.

- If you are checking a customer balance or budget check, save the value first and then add your transaction value to validate the expected result instead of validating a fixed expected value.
This topic contains information about how to troubleshoot the Regression suite automation tool (RSAT).

**Playback logs**

To troubleshoot issues that happen during playback of a test case, open the developer error log located at `[RSAT working directory]\[test case ID]\playback\[TestName]Log.txt`. The RSAT working directory is the directory specified in the RSAT settings dialog. Analyze the error message to determine the possible cause of a failure.

**NOTE**

For RSAT versions prior to 1.210, the log is located at C:\Users\[YourUserName]\AppData\Roaming\regressionTool\playback\[TestName]Log.txt.

**Generator logs**

To troubleshoot errors when generating test execution and parameter files, enable generator logs.

Open the `Microsoft.Dynamics.RegressionSuite.WindowsApp.exe.config` file under the RSAT installation folder (for example, C:\Program Files (x86)\Regression Suite Automation Tool), and change the value in the following element from `false` to `true`.

```xml
<add key="LogGeneration" value="true" />
```

After test execution files are generated, you can find the log file under `[RSAT working directory]\[test case ID]\generatorLogs`.

**NOTE**

For RSAT versions prior to 1.210, the logs are generated under C:\Users\[Username]\AppData\Roaming\regressionTool\generatorLogs.

**Authentication certificate and installation**

- The authentication certificate must be created and installed by an administrator on the same computer where RSAT is installed. If it is not created by an admin, you will encounter the following error message when you try to run a test case.

  Cannot access Finance and Operations environment. Verify your settings and make sure the environment is available.

- If you have used a previous version of RSAT on the same computer, close it and uninstall it before installing a new version.

**Screen resolution when using Internet Explorer**
If you have selected Internet Explorer as your browser, your desktop resolution should be set to 100% to run the tests successfully. To change the settings, use Windows Display settings > Scale and layout, as shown in the following image:

**Scale and layout**

Change the size of text, apps, and other items

100%

---

**Test playback errors**

**SOAP or HTTP request errors**

If you are testing against a Standard Acceptance Test Sandbox environment (Tier2) or any other multi-box environment, and you may receive one of the following errors when you run a test.

There was no endpoint listening at https://<yourURL>/soap.sandbox.operations.dynamics.com/Services/AxUserManagement/Service.svc/ws2007FedHttp that could accept the message. This is often caused by an incorrect address or SOAP action.

An error occurred while making the HTTP request to <Hostname>/Services/AxUserManagement/Service.svc/ws2007FedHttp. This could be due to the fact that the server certificate is not configured properly with HTTP.SYS in the HTTPS case. This could also be caused by a mismatch of the security binding between the client and the server.

Assuming that you have specified the correct SOAP hostname in the RSAT settings dialog box, run the following PowerShell scripts on your client computer where the test tool is installed.

```powershell
Set-ItemProperty HKLM:\SOFTWARE\Microsoft\.NETFramework\v4.0.30319 -Name SchUseStrongCrypto -Value 1 -Type dword -Force -Confirm:$false
if ((Test-Path HKLM:\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\v4.0.30319))  { Set-ItemProperty HKLM:\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\v4.0.30319 -Name SchUseStrongCrypto -Value 1 -Type dword -Force -Confirm:$false}
```

You can also manually set the registry keys.

**Cannot enumerate AX users error**

You may receive the following error when running a test case, or the error details may contain the following messages.

```plaintext
<Message>Could not enumerate AX users</Message> (InnerError)
```

To resolve this error, verify the **Admin user name** specified in the RSAT settings dialog box. The **Admin user**
Unsecured fault exception

If a test case inconsistently fails with the following error, this usually indicates an incomplete configuration of the authentication thumbprints on the AOS virtual machines.

```
<Message>An unsecured or incorrectly secured fault was received from the other party. See the inner FaultException for the fault code and detail.</Message>
<Message>At least one security token in the message could not be validated.</Message>
```

Typically, this error happens when the test environment has not been configured to trust the certificate that RSAT is using for authentication. (The certificate is identified by the thumbprint specified in your RSAT settings.) For example, the thumbprint could be missing in the wif.config file on the AOS virtual machine of the test environment. If you are running against a standard acceptance test environment (Tier 2 or higher), you might not have configured the authentication thumbprint on all of the AOS virtual machines. Make sure you properly add the thumbprint to the wif.config file on all of the AOS machines. For more information, see Configure the test environment to trust the connection.

We have also seen this error when there is a mismatch between the UTC time between the client computer (where RSAT is installed) and the Finance and Operations environment. This is a rare case and only happens if your administrator has incorrectly configured the UTC time. The client computer and Finance and Operations environment can be on different time zones; however, the UTC time on both environments must match because the authentication mechanism relies on this comparison.

Google Chrome Browser

The Google Chrome browser may not work with the Regression suite automation tool due to your Active Directory security settings. In this case, change your RSAT settings to use the new Microsoft Edge or Internet Explorer.

Validating blank dates

If your test case requires validation that a certain control of type Date/Time is blank, you can insert the following value into the Excel cell corresponding to this control: “01/01/1900”.

Azure DevOps connectivity

You might you see this error when you select the desired Azure DevOps project in RSAT settings: "The structure path <iteration path> is not valid. Verify your settings and try again". To resolve this error, open the project in Azure DevOps and navigate to the Test Plans. Verify the iteration path defined for each test plan. If the iteration path is similar to what is shown in the error, remove the existing iteration path and add a new one for the test plan and save.
Lifecycle Services (LCS) solution packages for Microsoft AppSource are partner-designed and developed solutions that can be automatically deployed on Microsoft Azure to deliver an end-to-end solution, using industry and vertical-specific content.

This topic points to resources that will help you understand the requirements for creating LCS solutions for Microsoft AppSource. The requirements for creating an LCS solution package fall into the following groups.

- App validation
- Code migration
- Database backup and Data packages
- Business process models
- Methodologies
- Marketing
This topic explains how to create and modify methodologies in Microsoft Dynamics Lifecycle Services (LCS). It also provides information about the requirements for methodologies.

Methodology requirements

The first two phases of your methodology are the Learn and Consume phases from the LCS Solutions Consumption methodology.

During the Learn phase, your product description must be aligned with your business process library, the descriptions or summaries in your marketing material, and the functionality that is supported in the current version of your solution. The Learn phase includes the following tasks:

- **Product description** – Take advantage of your marketing description by adding it to the LCS methodology.
- **Get an overview** – Include content for your solution, such as information about the features and architecture.
- **How to get help** – Include numbers, contacts, or direct links to your company’s website, for people who help build and maintain the solution.

The Consume phase is optional. It includes tasks that are required in order to complete Conference room pilot 1 (CRP1).

After the Learn and Consume phases, you can add any other steps that are required in order to implement your solution. You can either modify the phases and tasks so that they are aligned with your solution, or use your company’s implementation methodology.

Create a new methodology

1. On the LCS home page, select the **Manage methodologies** tile.
2. Select the **New methodology** button (the plus sign [+]).
3. Set values for the fields, and then select **Confirm**.
4. Create phases and tasks.
5. Add any linked tools and resources that are required.

Edit a methodology

1. On the LCS home page, select the **Manage methodologies** tile.
2. Select the methodology to edit.
3. Select the **Edit methodology** button (the pencil symbol), and edit the methodology.

Edit a project's methodology

Follow these steps to edit a methodology in a specific project only.

1. On the project's home page, select the ellipsis (...) button above the project phases, and then select **Edit methodology**.
2. Edit the phases and tasks.
3. Edit the linked tools and resources, if any changes are required.
Change a project’s methodology

1. On the project’s home page, click the ellipsis (…) button above the project phases, and then click Change methodology.

2. In the dialog box that appears, set the Do you want to keep the existing phases and tasks? option. If you set this option to Yes, the phases and tasks from the methodology that you select in step 3 are added to the end of the current methodology. If you set the option to No, the current methodology is replaced with the methodology that you select in step 3.

3. Select a methodology to use with your project.

4. Select Confirm.

Additional resources

Requirements for publishing apps on AppSource
Create a business process library

There are two ways to create a BPM library. You can create a new library that has no lines or task recordings, or you can copy an existing library.

1. In Microsoft Dynamics Lifecycle Services (LCS), open a project, scroll to the right until you see the **More tools** section, and then select the **Business process modeler** tile. The **Business process libraries** page that appears has three sections, one for each type of library:
   - **My libraries** – Business processes that users have created or added.
   - **Corporate libraries** – Custom business processes that someone in your organization has uploaded.
   - **Global libraries** – Cross-industry standard business processes.

2. To create a new library, right-click any library, and then, in the lower-left corner of the window, select **Create**. To copy an existing library, right-click that library, and then, in the lower-left corner of the window, select **Copy**.
Update, publish, or delete a business process library

You can change the name or description of a library, publish a library so that other people in your organization can view it, or delete a library.

- On the **Business process libraries** page, right-click the library to update, publish, or delete. Then, in the lower-left corner of the window, select **Edit**, **Publish**, or **Delete**.
  - **Edit** lets you change the name and description of the library.
  - **Publish** creates a copy of the library under **Corporate libraries**. This library will be available to everyone in your organization.
  - **Delete** deletes the library and any information that is stored in it.

Modify a business process library

Follow these steps to change or update the business process lines or hierarchy in your business process library.

1. Select and open the library to update.
2. In the left pane, under **Core views**, select **Author and edit**.
3. Follow one or more of these steps to make the required changes:
   - To rearrange the hierarchy of the business process lines, drag the processes in the hierarchy.
   - To delete a node, select a business process in the center pane, and then select **Delete** in the right pane.
   - To create a new node, drag the **New Business Process** flag at the top of the center pane to the place in the hierarchy where the new node should appear.
To import business processes from existing business process libraries, follow these steps:

a. Select Import in the center pane.

b. On the right side of the page that appears, select the library to import business processes from.

c. Drag the required business processes into the hierarchy on the left side of the page.

View and modify task recordings in a business process library

1. Open the library that you want to work in.

2. Navigate to the business process line that has a task recording associated with it, and select the link.

3. Drag the tiles under Activities to manually modify the flow chart.

4. To upload a Microsoft Visio diagram of the modified flow chart, select the Visio tab.

Structure a business process library

There are two sections in business process libraries: Core Business Processes and Support Processes. The Core Business Processes section should include all custom business processes for your solution. All customizations and functionality should be covered in end-to-end scenarios. Task recordings should be created for all processes in this section. Import American Productivity & Quality Center (APQC) processes that are relevant to your solution into the Support Processes section. This section should not include any custom business processes. You don't have to create task recordings for processes in this section. Your business process library should be aligned with the descriptions and summaries in your methodology and your marketing material.

Create a task recording and associate it with a business process

Task recordings should be created in an environment that has your custom data and customizations. For reference information about Task recorder, see Task recorder resources.

Create a task recording

1. In your application, select the Settings button in the upper-right corner, and then select Task recorder.

2. Select Create recording.

3. Enter a name and description for the recording.

4. Perform the task that you want to record.
5. When you’ve completed the task, select **Stop**.

6. If you want to upload your new task recording to LCS, continue to the next section. To save your recording to your local computer or convert the recording to a Microsoft Word document, select the appropriate option.

**Associate a task recording with a business process**

1. Select **Save to LCS**.
2. Select a business process library.
3. Select the business process line to associate with the recording.
4. Select **OK**.

You can now view the task recording in the business process library in LCS.

**Set up and use task guides**

Task recordings can be played as task guides. Task guides are used to guide users through the steps for completing business processes. Before you set up and use task guides, create task recordings, and save them to a business process library in LCS.

**Set up task guides**

1. In your application, select **System administration > Setup > System parameters**.
2. On the **Help** tab, select the LCS project where the business process library that you want to work with is stored.
3. Select the business process libraries that have the task recordings that you want to play as task guides.
4. Adjust the order of the business process libraries as you require. The order of the business process libraries determines the order that the task guides appear in. Therefore, the task guides for the first business process library appear first when a user uses the task guide functionality.

**Use task guides**

1. In your application, open the page where you want to run the task guide.
2. In the upper-right corner, select the **Settings** button, and then select **Task recorder**.
3. Select a task guide.
4. Select **Start Task guide**.
5. Follow the steps in the task guide. If you select **Unlock**, you can work without following the task guide.
6. When you’ve finished, select **Stop Task guide**.

**Additional resources**

Requirements for publishing apps on AppSource](lcs-solutions-app-source.md)
To complete your solution package, the first step is to upgrade your code by using the best practices in Migrate and Create Finance and Operations Apps Solutions in Microsoft Dynamics Lifecycle Services (LCS). After that step is completed, you must run the Customization Analysis Report (CAR). This report analyzes your customization and extension models, and runs a predefined set of best practice rules.

To generate the CAR, run the following command on a development environment.

```bash
xppbp.exe -metadata=<local packages folder> -all -model=<ModelName> -xmlLog=C:\BPCheckLogcd.xml -module=<PackageName> -car=<reportlocation>
```

Here is an example of this command.

```bash
xppbp.exe -metadata=C:\Packages -all -model=MyAppSuiteCustomizations -xmlLog=C:\temp\BPCheckLogcd.xml -module=ApplicationSuite -car=c:\temp\CARreport.xlsx
```

The xppbp.exe file is located in `c:\packages\bin` or `I:\AosService\Packages\LocalDirectory\bin`. You must resolve any warnings or errors that appear on the Issues tab of the report. You must then submit a copy of the CAR to Microsoft before your validation meeting. For more information, see Customization Analysis Report (CAR). For information about issues and exceptions, see the Customization Analysis Report: Exceptions and known issues post on the Dynamics 365 Community blog.

**Extensibility**

In Microsoft Dynamics 365 for Finance and Operations version 8.0 (April 2018), all product models are sealed. Therefore, only extension-based customizations are currently supported. For more information about extensibility, see Extensibility.

The first step in completing your solution package is to upgrade your code using the best practices in Migrate and Create Finance and Operations Apps Solutions in LCS. After this step is complete, you must run the Customization Analysis report. This report analyzes your customization and extension models, and runs a predefined set of best practice rules.

To generate the Customization Analysis report (CAR), run the following command on a development environment.

```bash
xppbp.exe -metadata=<local packages folder> -all -model=<ModelName> -xmlLog=C:\BPCheckLogcd.xml -module=<PackageName> -car=<reportlocation>
```

Here's an example of how this command might look.

```bash
xppbp.exe -metadata=C:\Packages -all -model=MyAppSuiteCustomizations -xmlLog=C:\temp\BPCheckLogcd.xml -module=ApplicationSuite -car=c:\temp\CARreport.xlsx
```

The xppbp.exe file is located in `c:\packages\bin` or `I:\AosService\Packages\LocalDirectory\bin`. Any warnings or
errors that appear on the Issues tab of the report must be resolved. A copy of the CAR report must be submitted to Microsoft prior to your validation meeting. For more information, see Customization Analysis Report (CAR) or refer to the Dynamics Community blog for issues and exceptions.

Additional resources

Requirements for publishing apps on AppSource

Develop and customize home page
This topic provides information about the requirements that are used to verify that custom code meets Microsoft guidelines, and that a solution package can be successfully bundled and delivered in a Finance and Operations apps environment.

Microsoft requires specific reviews in order to validate the following requirements:

- A partner’s custom code meets Microsoft guidelines.
- A Microsoft Dynamics Lifecycle Services (LCS) solution package can be successfully bundled and delivered.
- Core independent software vendor (ISV) business scenarios can be transacted.

Currently, partners must demonstrate that these requirements have been met by doing test deployments and then sharing the results with Microsoft. No code will be deployed on a customer environment that Microsoft hasn’t validated. Partners must complete the following curation artifacts and tests:

- Code analysis report (CAR)
- Business process modeler (BPM)/test scripts
- Business database backup
- Project name and description
- Data packages
- Methodology
- Binaries (optional)
- Deployable packages
- Models (code and tests)
- Marketing content

### Curation meeting

The following table describes the steps that must be completed before the validation meeting.

<table>
<thead>
<tr>
<th>PHASE</th>
<th>STEP</th>
<th>ACTIVITY</th>
<th>PROCESS STEPS</th>
<th>SUCCESS CRITERIA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Validate code.</td>
<td>Run all customer model files by using the CAR tool, and then generate the report.</td>
<td>Successfully create a CAR without any localization, accessibility, performance, or security issues.</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>Verify user experience (UX) guidelines.</td>
<td>Follow UX guidelines to implement the workspace correctly.</td>
<td>Reference best practice information in the Migrate and Create methodology section of LCS.</td>
</tr>
<tr>
<td>PHASE</td>
<td>STEP</td>
<td>ACTIVITY</td>
<td>PROCESS STEPS</td>
<td>SUCCESS CRITERIA</td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td>----------</td>
<td>---------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>Validate the solution package in LCS.</td>
<td>Create a solution package in LCS that includes all the required artifacts.</td>
<td>A solution package that has all the required artifacts has been published in LCS, and a globally unique identifier (GUID) has been created for the solution.</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>Deploy an environment.</td>
<td>Deploy a standard environment that has partner code, based on the package contents (code, binaries, and configuration).</td>
<td>Successfully deploy at least one Finance and Operations environment without any errors. The environment configuration (including components and the configuration) is the same as the partner’s reference environment. A user can successfully sign in to this environment without any errors.</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>Configure and deploy data.</td>
<td>Deploy partner-supplied data in the environment without any errors.</td>
<td>Demonstrate that partner-supplied master and reference data was successfully pushed into the environment without any errors.</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>Do a sanity check.</td>
<td>After data has been loaded into the environment, users should be able to complete business transactions (as defined in the scope of the solution).</td>
<td>Users can sign in to the data-loaded environment without any errors. Business transactions can be completed, as defined in the package scope, without any errors.</td>
</tr>
</tbody>
</table>

**Detailed curation requirements**

The following table provides more information about each curation requirement.

<table>
<thead>
<tr>
<th>REQUIREMENT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAR</td>
<td>All major issues that the CAR highlights should be addressed after you upgrade. The CAR must be submitted to Microsoft before the validation meeting.</td>
</tr>
<tr>
<td>BPM/test scripts</td>
<td>All task recordings should be completed for the industry vertical that the solution package is designed for and should include end-to-end scenarios.</td>
</tr>
<tr>
<td>REQUIREMENT</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Business database backup</td>
<td>A business database of your upgraded environment and best practice configurations should be loaded into the Asset library in LCS.</td>
</tr>
<tr>
<td>Project name and description</td>
<td>The project name and description should be incorporated into the beginning of the implementation methodology for the solution package.</td>
</tr>
<tr>
<td>Data packages</td>
<td>All data packages should be loaded into LCS before the validation meeting. Create data entities for any additional custom fields or tables for your custom functional features. You should be able to modify the data packages and load them into an empty environment, and then consume the data packages in Data Management Framework.</td>
</tr>
<tr>
<td>Methodology</td>
<td>The methodology should incorporate an overview of the product. A guided experience to Conference Room Pilot 1 (CRP1) and any other implementation methodology that is specifically tailored to your solution are optional.</td>
</tr>
<tr>
<td>Binaries (optional)</td>
<td>Incorporate any required binary files.</td>
</tr>
<tr>
<td>Deployable packages</td>
<td>Incorporate the deployable packages that are required in order to bring your custom features and functionality into your environment.</td>
</tr>
<tr>
<td>Models (code and tests)</td>
<td>Incorporate any model files that are required for your solution.</td>
</tr>
<tr>
<td>Marketing content</td>
<td>Add your marketing content, such as logos, descriptions, and screen shots of your solution package. The solution logo should be app-specific and should not include your company name. The description should be aligned with your custom business processes.</td>
</tr>
</tbody>
</table>

**Update and maintenance requirements**

If you have a curated solution that is published on AppSource, you must keep the solution up to date. After each major Spring and Fall release, you will have eight weeks to upgrade your code. You must update and test the following artifacts:

- CAR
- Models (code and tests)
- Deployable packages
- BPM/test scripts
- Data packages
- Business database backup

**Maintenance process steps**

<table>
<thead>
<tr>
<th>PHASE</th>
<th>NUMBER</th>
<th>ACTIVITY</th>
<th>PROCESS STEPS</th>
<th>SUCCESS CRITERIA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHASE</td>
<td>NUMBER</td>
<td>ACTIVITY</td>
<td>PROCESS STEPS</td>
<td>SUCCESS CRITERIA</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>--------------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Validate customer code.</td>
<td>Run all customer model files by using the CAR tool, and generate the report.</td>
<td>Successfully create a CAR without any localization, accessibility, performance, or security issues. All major issues that the CAR highlights should be addressed after you've upgraded to the latest major release. The CAR must be submitted to Microsoft within eight weeks after each major Spring and Fall release.</td>
</tr>
</tbody>
</table>

### Additional resources

[Requirements for publishing apps on AppSource](#)
A data package for a Dynamics 365 Finance and Operations app can consist of one or many data entities. A typical data package consists of a group of entities for a specific task, process, or function. For example, the data entities that are required for general ledger setup might be part of one data package. The format of a data package is a compressed file that contains a package manifest, a package header, and any additional files for the data entities that are included.

Before you create your data package, plan out what it should include. In this way, you help guarantee that the correct entities, entity sequence, and fields are included. You create a data package by using the Data management workspace in your application. Follow these steps to create a data package.

1. In the application, select **System administration** > **Workspaces** > **Data Management IT**.
2. Select the **Export** tile, and then, in the **Name** field, enter **Data project**.
3. In the **Target data format** field, select the format for the export. The data formats that are available include comma-separated value (CSV) format and Microsoft Excel format.
4. In the **Entity name** field, enter or select an entity. You can add multiple entities, but you must add each entity separately.
5. In the **Select fields** field, select a field setting, and then click **Add entity** to add the entity to the project.
6. Repeat steps 4 and 5 to add more entities to the project. **Note:** As you add each entity to the project, a tile appears that contains the entity's name and two buttons: **View map** and **Filter**. To automatically create a data package when you export the data project, set the **Generate data package** option to **Yes**. If you don't set this option, you can create a data package at the time of export.
7. On the Action Pane, click **Export**.
8. Click **Download**. The package is saved to the **Downloads** folder of the computer where the browser session is running. When you work with data packages, you must plan for and consider any prerequisites for the entities that will be included in the packages. For example, the customer groups are required in order to create customers. Therefore, you should either import the customer groups into a package before you import customers, or sequence customer groups within a data package that will be completed before customers are imported. For example, in the following illustration, sequencing is set within the data package. As you can see, the Customer groups entity and Customers entity are part of the Customers data project.
To automatically create a data package when you export the data project, set the **Generate data package** option to **Yes**. If you don't set this option, you can create a data package at the time of export.

9. On the Action Pane, select **Export**.

10. Select **Download**. The package is saved to the **Downloads** folder of the computer where the browser session is running. When you work with data packages, you must plan for and consider any prerequisites for the entities that will be included in the packages. For example, customer groups are required in order to create customers. Therefore, you should either import the customer groups into a package before you import customers, or sequence customer groups within a data package that will be completed before customers are imported. For example, in the following illustration, sequencing is set in the data package. As you can see, the Customer groups entity and Customers entity are part of the Customers data project.

11. On the Action Pane, select **Entity sequence** to open the **Definition group entity sequence** page. Based on the current setup, the Customer groups entity and Customers entity are run at the same level. However, this sequence might not be ideal.
12. To create a better sequence, select the **Customers** entity, and then update the value of the **Execution unit** field from 1 to 2. This change helps guarantee that customer groups are imported before the Customers entity is run.

Microsoft Dynamics Lifecycle Services (LCS) contains multiple base data packages that you can use to reduce the implementation time. These packages contain the elements that are required in each module/area in order to meet the minimum requirements. For advanced business processes, you might have to add more entities to the list of packages. The data packages that Microsoft publishes on LCS use a numbering sequence that is based on the module, data type, and sequence. Here is an example:

- **Module/area number**

<table>
<thead>
<tr>
<th>Module</th>
<th>Module Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>System administration</td>
<td>01</td>
</tr>
<tr>
<td>General ledger</td>
<td>03</td>
</tr>
<tr>
<td>Public Sector</td>
<td>04</td>
</tr>
<tr>
<td>HRM</td>
<td>05</td>
</tr>
<tr>
<td>Accounts payable</td>
<td>10</td>
</tr>
<tr>
<td>Accounts receivable</td>
<td>11</td>
</tr>
<tr>
<td>Budgeting</td>
<td>12</td>
</tr>
<tr>
<td>Cash and bank management</td>
<td>13</td>
</tr>
<tr>
<td>Compliance and internal controls</td>
<td>14</td>
</tr>
<tr>
<td>Cost accounting</td>
<td>15</td>
</tr>
<tr>
<td>Fixed assets</td>
<td>16</td>
</tr>
<tr>
<td>Inventory management</td>
<td>19</td>
</tr>
<tr>
<td>Master planning</td>
<td>20</td>
</tr>
<tr>
<td>Organization administration</td>
<td>21</td>
</tr>
<tr>
<td>Payroll</td>
<td>22</td>
</tr>
<tr>
<td>Procurement and sourcing</td>
<td>23</td>
</tr>
<tr>
<td>Product information management</td>
<td>24</td>
</tr>
<tr>
<td>Production control</td>
<td>25</td>
</tr>
<tr>
<td>Project management and accounting</td>
<td>26</td>
</tr>
<tr>
<td>Retail</td>
<td>27</td>
</tr>
<tr>
<td>Sales and marketing</td>
<td>28</td>
</tr>
<tr>
<td>Service management</td>
<td>29</td>
</tr>
<tr>
<td>Trade allowance management</td>
<td>31</td>
</tr>
<tr>
<td>Transportation management</td>
<td>32</td>
</tr>
<tr>
<td>Travel and expense</td>
<td>33</td>
</tr>
<tr>
<td>Warehouse management</td>
<td>34</td>
</tr>
</tbody>
</table>

- **Data type numbering**

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Data Type Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup</td>
<td>1</td>
</tr>
<tr>
<td>Master</td>
<td>4</td>
</tr>
<tr>
<td>Transaction</td>
<td>8</td>
</tr>
</tbody>
</table>

- **Numbering format**
The names of data packages include the numbering format, which is followed by the module abbreviation and then a description. For example, the following illustration shows the General ledger data packages.

<table>
<thead>
<tr>
<th>Numbering Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module #. Data Type Reference .001 (Sequence Number)</td>
</tr>
<tr>
<td>01.1.001</td>
</tr>
<tr>
<td>01.1.002</td>
</tr>
<tr>
<td>01.4.001</td>
</tr>
<tr>
<td>01.4.002</td>
</tr>
<tr>
<td>03.1.001</td>
</tr>
</tbody>
</table>

Process data packages

A process data package (PDP) consolidates Data import/export framework (DIXF) data packages into a unified bundle. The PDP is then used to configure a business process or a group of business processes in one business process library. Together, DIXF data packages, dependencies between those packages, and business processes that require the packages for their configuration make up a PDP. This section describes how to create a PDP for your LCS solution package. To create a PDP, you must have the following items and knowledge:

- An implementation business process library for the solution in your working LCS project.
- DIXF data packages that have configurations that follow best practices. These packages should include master and reference data for the whole solution.
- An understanding of the dependencies between the data in the packages.
- An understanding of the data dependencies between the data packages.

Follow these steps to create a PDP.

1. In LCS, select the Asset library tile.
2. Select Data package as the asset type, and then select the plus sign (+) to add a data package.

Follow these steps to consolidate the data packages that you uploaded to LCS into a single PDP.

1. In LCS, select the Asset library tile.
2. Select Process data package as the asset type, and then select the plus sign (+) to add a PDP.
3. Enter a name and description for the PDP, and then select Confirm.
4. For step 1, “Add business process library,” select the implementation business process library that your solution is built for, and then select Continue.
5. For step 2, “Add data packages,” select Edit, and then add the data packages that are required in order to configure the system and enable the business process library transactions to be run. When you've finished adding data packages, select Select. The order in which data packages in the PDP are loaded into the system might be important. For example, before a chart of accounts (COA) can be loaded, the legal entity setup and currencies are required. Those data entities are in 01.1.002 System Setup. Therefore, 03.1.1001 COA Setup depends on 01.1.002 System Setup. In the next step, you must make a note of this dependency.
6. For step 3, “Add data package dependencies,” select a data package, and then select Edit. Select the dependent data packages that must be loaded into your target environment before the data package that you selected, and then select Select.
7. For step 4, “Associate business process to data packages,” select the business process that should be used to configure the process nodes.
Consume a PDP

**IMPORTANT**

For the PDP consumption requirement, you have the option to consume data packages directly via the Data Management Framework in your environment. However, note that only the consumption of data packages via the LCS PDP tool is optional. You must still create the PDP and upload it to your Asset library.

The Consume flow lets you review a business process, and apply the configuration and data that are required in order to implement the business process in your environment. To consume PDPs, you must have the following items:

- An implementation business process library for the solution in your working LCS project
- PDPs
- A target environment that includes an existing legal entity

Follow these steps to consume the PDP:

1. In LCS, select the Asset library tile.
2. Select Process data packages as the asset type, and then select Consume.
3. The Consume process data package page shows a list of the existing PDP assets. Select the plus sign (+) to create a new Consume PDP.
4. Enter a name for the Consume PDP, select the PDP that you created and saved in the Asset library, select a target environment, and then select Create.
5. The Consume PDP that you created appears in the list of PDP assets. Select the package.

**NOTE**

The asset that you created can be consumed only in the target environment that was linked to it.

Review and approve BPMs

- For step 1, “Review business process,” review the business process models (BPMs), and then select Mark as reviewed. The review status is updated for all dependent processes, and a green bar appears to the right of them. The Reviewed by and Completed on fields are also updated for each business process.

Review and approve data packages that are associated with a BPM

1. In the BPM library that you just reviewed, in the left pane, select Approve the data packages.
2. Select the business process that is associated with the data packages. The data packages appear in the right pane, under Process details. In the right pane, select Review and Approve.
3. On the **Consume process data package** page, select a package, and then select **Download** to download the data package. Save the package locally. You can then update the data files in the data package with data that is specific to your target environment. When you’ve finished updating the data files, select **Upload data package** to upload your changes.

4. After the data package has been updated and completed, select **Approve**. The status is changed to **Approved**. You can approve as many data packages as you require.

5. Select the **Back** button, select 2 **Approve the data packages** again, and then select the business process. You should see the **Approved** status in the right pane, under **Process details** > **Dependent packages**.

**Apply a process data package**

1. In the BPM library, in the left pane, select 3 **Apply process data package**.
2. Select the business process, and then, in the right pane, select **Apply Data Packages**.
3. On the **Consume process data package** page, select a package, and then select **Apply**.
4. Select the destination company in the target environment that is linked to the PDP, and then select **Apply**.

   - For step 1, ”Review business process,” review the business process models (BPMs), and then click **Mark as reviewed**. The review status is updated for all dependent processes, and a green bar appears to the right of them. The **Reviewed by** and **Completed on** fields are also updated for each business process.

**Review and approve data packages that are associated with a BPM**

1. In the BPM library that you just reviewed, click 2 **Approve the data packages** in the left pane.
2. Select the business process that is associated with the data packages. The data packages appear in the right pane, under **Process details**. Click **Review and Approve** in the right pane.
3. On the **Consume process data package** page, select a package, and then click **Download** to download the data package. Save the package locally. You can then update the data files in the data package with data that is specific to your target environment. When you’ve finished updating the data files, click **Upload data package** to upload your changes.
4. After the data package has been updated and completed, click **Approve**. The status is changed to **Approved**. You can approve as many data packages as necessary.
5. Click the **Back** button, click 2 **Approve the data packages** again, and then select the business process. You should see the **Approved** status in the right pane, under **Process details** > **Dependent packages**.

**Apply a process data package**

1. In the BPM library, click 3 **Apply process data package** in the left pane.
2. Select the business process, and then, in the right pane, click **Apply Data Packages**.
3. On the **Consume process data package** page, select a package, and then click **Apply**.
4. Select the destination company in the target environment that is linked to the PDP, and then click **Apply**.

**View the data package history**

- On the **Consume process data package** page, select a package, and then select **History**. You can review the status of the data package. The available information includes the target environment, company, package
name, start and end times, status by data entity, and overall status of the data package. To see the details of any errors that occurred, you can sign in to the target environment.

**Additional resources**

Requirements for publishing apps on AppSource
A backup of the Finance and Operations apps database is required for your Microsoft Dynamics Lifecycle Services (LCS) solution package. When you back up the database, you must include the master, reference, and transactional data that is specific to your solution and industry. This data will be used for your pre-sales demo deployments.

On demo or development environments, the database is typically named AXDBRain. Your database backup should be no larger than 15 gigabytes (GB). Otherwise, a time-out error might occur when you try to upload the database to the Asset library in LCS.

To compress your database backup, in Microsoft SQL Server Management Studio, on the Back Up Database page, in the Set backup compression field, select Compress backup.

On demo or development environments, the database is typically called AXDBRain. Your database backup should be no larger than 15 gigabytes (GB). If your database is larger, a time-out error may occur when you try to upload the database to the Asset library in Lifecycle Services (LCS).

To compress your database backup, in SQL Server Management Studio, on the Back Up Database page, in the Set backup compression field, select Compress backup.
Additional resources

Requirements for publishing apps on AppSource
Local and regional deployments

If your government regulations require data to be stored differently or serviced differently than is required for other countries/regions, there might be country/region requirements you must consider during deployment. Consider the following resources that might be relevant to you:

Finance and Operations apps operated by 21Vianet in China

Localization and regulatory features

Finance and Operations apps include functionality for the country/regions documented in the Product localization and translation availability guide. This functionality is enabled based on the primary address of the active legal entity.

This topic includes lists of resources that can help you do the following:

- Learn more about developing country/region-specific solutions.
- Get country/region specific updates.
- Submit and review regulatory alerts.
- Learn how to use country/region specific functionality.

Developing localized solutions

The following resources provides guidance and information that can help developers and ISVs who are creating country/region-specific customizations or are creating a solution for a country that Microsoft does not support.

- Separation of localization models
- Apply country/region context
- Regulatory certification information in feature titles
- Classification of localization features
- Country Codes - ISO 3166

Regulatory updates and communication

The following resources provide information about planned and new localization features.

Regulatory updates

- Regulatory updates
- Issue search in Lifecycle Services (LCS) (Updated daily)

Communication and alerts

- Regulatory watch and communication of regulatory updates
- Submit alerts about country/region-specific regulatory features

Dynamics 365 release plans

The Dynamics 365 release plans provide descriptions of new and enhanced capabilities that are planned for Dynamics 365 business applications and application platforms.

Finance and Operations apps what's new

The What’s new or changed in Finance and Operations home page lists the features that are included in specific releases of the Finance and Operations apps.
Electronic reporting

The Electronic reporting (ER) tool allows you to configure formats for electronic documents in accordance with the legal requirements of various countries/regions. ER lets you manage these formats during their lifecycle. For more information, refer to one of the following topics:

- Electronic reporting (ER) overview
- Manage the Electronic reporting (ER) configuration lifecycle
- Create Electronic reporting (ER) configurations
- Extend the list of Electronic reporting (ER) functions
- Electronic reporting (ER) destinations
- Download Electronic reporting configurations from Lifecycle Services
- Import Electronic reporting (ER) configurations
- Configure Electronic reporting (ER) to pull data into Power BI
- Generate electronic documents and update application data by using ER

Task guides

Task guides are available from the product help pane and they provide a guided walk-through of key business processes. You can open a task guide to read the steps of a business process or you can play a task guide to walk through a business process and enter data.

To find task guides, navigate to a page in the application and click Help. Task guides that use the page are listed in the help pane. You can also use the help pane to search for task guides by title.

To learn more, see Help system.

Country/region specific help content

- Australia
- Austria
- Belgium
- Brazil
- China
- The Czech Republic
- Estonia
- Europe
- France
- Germany
- Hungary
- India
- Italy
- Japan
- Latvia
- Lithuania
- Mexico
- Malaysia
- Netherlands
- Norway
- Poland
- Russia
- Saudi Arabia
- Singapore
- Spain
- Sweden
- Switzerland
- Thailand
- United Kingdom
- United States
As part of the requirements for LCS solutions for localization and translation, localization features must be classified as either regulatory or competitive in the business process modeler (BPM) library in Microsoft Dynamics Lifecycle Services (LCS). This article explains the difference between these two types of feature and shows how the feature type is used in the title of the feature.

Localization features must be classified as either regulatory or competitive in the Business process modeler (BPM) library in Microsoft Dynamics Lifecycle Services (LCS). The following definitions will help you distinguish the two types of features:

- **Regulatory features** – Organizations that do business in a particular country/region must comply with country/region-specific laws and regulations as they handle their daily business transactions and operations, and meet their legal obligations for activities that are conducted in that country/region. These laws and regulations are enforced, and non-adherence can lead to severe consequences for an organization that does business in that country/region.

- **Competitive features** – This category includes all other features that aren’t considered regulatory features according to the preceding definition.

When the BPM library is constructed, the various feature types should be distinguished through the title of the localization solution feature. The label for this title will conform to the following naming convention that indicates the country/region and feature type through prefixes, as shown in the following table.

<table>
<thead>
<tr>
<th>FEATURE TYPE</th>
<th>FORMAT</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regulatory</td>
<td>XX-REG-Feature title</td>
<td>PT-REG-Direct sales tax report</td>
</tr>
<tr>
<td>Competitive</td>
<td>XX-COMP-Feature title</td>
<td>PT-COMP-Country bank payment format</td>
</tr>
</tbody>
</table>

The following table explains the components of the naming convention.

<table>
<thead>
<tr>
<th>COMPONENT NAME</th>
<th>FORMAT</th>
<th>DESCRIPTION</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Country/region code</td>
<td>XX</td>
<td>The two-letter ISO country/region code (from the ISO 3166 standard)</td>
<td>PT (= Portugal)</td>
</tr>
<tr>
<td>Feature type</td>
<td>REG or COMP</td>
<td>The type of feature</td>
<td>REG</td>
</tr>
<tr>
<td>Feature name</td>
<td>Text</td>
<td>A short feature title that describes what the feature is used for</td>
<td>Direct sales tax report</td>
</tr>
</tbody>
</table>


### Additional resources

- [Globalization resources](https://docs.microsoft.com/en-us/dynamics365/lifecycle-services/globalization-overview)
As part of the requirements for LCS solutions for localization and translation, localization ISV solution providers must implement all country-specific or region-specific functionality so that it can be controlled by country/region context. This article describes how to apply country/region context to meet these requirements. In this article you can find information how you should use country context property and what application objects control user interface elements.

### Country/region-specific functionality

You use country/region-specific functionality to help meet the legal, regulatory, and business requirements of individual geographies. A geography is any country or region that is identified by an International Organization for Standardization (ISO) country or region code. The following table highlights the main elements that you use to configure country/region-specific functionality.

<table>
<thead>
<tr>
<th>ELEMENT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| Controlled entity | The controlled entity is a UI element that is hidden or shown, depending on whether its country/region context matches the country/region context of the controlling entity. To enable menus, menu items, and form controls that are based on country/region context to be hidden, a controlled entity includes a `CountryRegionCodes` property on some elements. You use this property to specify the country or region where the element is shown. You find the `CountryRegionCodes` property on the following Application Object Tree (AOT) elements:  
  - Extended data type  
  - Menu and menu items  
  - Enum and enum value  
  - Table and table field  
  - Data entity and data entity field  
  - View and view field  
  - Map and map field  
  - Form control  
  - Tile |
| Controlling party | The controlling party's role is used to determine whether country/region-specific functionality or UI elements are enabled. The controlling party is defined by the Organization model. Examples include legal entity, customer, vendor, bank, or worker. By default, the legal entity is used as the controlling party. If the country/region context of the controlling party matches the country/region context of the controlled entity, the functionality or UI elements are enabled. You set the country/region context of the controlling party. Any controlled entities that have a matching country/region context are shown. |

### Using the CountryRegionCodes property

You create country/region context on a controlled entity by setting the ISO code value on the
**CountryRegionCodes** property. You can find the list of ISO country and region codes on the ISO website. The values of the **CountryRegionCodes** property are compared to the country/region context of the controlling party. If the values match, the element is shown. Otherwise, it's hidden.

**TIP**
To add more than one ISO country and region code to the **CountryRegionCodes** property, use a comma-separated list.

**Using the legal entity as the controlling party**

The **Country/region** value in the primary address of the legal entity determines the country/region context of the controlling party. The default value of the **Country/region** field is the locale of the system. The following illustration shows how to set the primary address of the legal entity.

![Primary address of the legal entity](image)

**Setting another party as the controlling party**

You can use another party, such as a customer, bank, or vendor, as a controlling party. For example, you can enable targeted functionality for customers of a specific country/region or require specific validation of vendors from a specific country/region. To set the controlling party, use the **CountryRegionContextField** property of the form, control, or other element. This property lets you select the entity that is the controlling party. The default value is the legal entity. The following illustration shows how to set the **CountryRegionContextField** property for a field.

![Setting another party as the controlling party](image)
In this example, the customer becomes the controlling entity. The customer’s address is compared with the value of the `CountryRegionCodes` field to determine whether the `GermanSpecificSetting` field is displayed.

**Additional resources**

ISO codes
As part of the requirements for LCS solutions for localization & translation, localization ISV solution providers must include details about any regulatory certifications that the solution requires in order to be legally compliant for sale in the intended market. This article shows how information about certifications is used in the title of the feature.

Regulatory certification can take various forms, from data privacy to certification of compliance with specific regulations. However, one thing that all regulatory certifications have in common is that they are required by the laws and regulations of the country/region of operation. These certifications are enforced, and non-adherence can lead to severe consequences for an organization that does business in that country/region. When the business process library is constructed, regulatory certifications must be identified as regulatory requirements in the Microsoft Dynamics Lifecycle Services (LCS) Business process modeler (BPM) library through the title of the localization solution feature. The label for this title will conform to the following naming convention that indicates the country/region and certification type through prefixes, as shown in the following table.

<table>
<thead>
<tr>
<th>FEATURE TYPE</th>
<th>FORMAT</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regulatory</td>
<td>XX-REG-Certification for xx</td>
<td>PT-REG-Certification for Fiscal printers</td>
</tr>
</tbody>
</table>

The following table explains the components of the naming convention.

<table>
<thead>
<tr>
<th>COMPONENT NAME</th>
<th>FORMAT</th>
<th>DESCRIPTION</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Country/region code</td>
<td>XX</td>
<td>The two-letter ISO country/region code (from the ISO 3166 standard)</td>
<td>PT (= Portugal)</td>
</tr>
<tr>
<td>Feature type</td>
<td>REG</td>
<td>The type of feature</td>
<td>REG</td>
</tr>
<tr>
<td>Certification name</td>
<td>Text</td>
<td>A short title that describes the certification and its application</td>
<td>Certification for Fiscal printers</td>
</tr>
</tbody>
</table>

In the BPM business process library, certifications should be located under **APQC level 8.0 Manage Financial Resources (10009)**.

**Example**

- APQC level 8.0 Manage Financial Resources (10009)
  - PT-REG-Certification for Fiscal printers

For more information about BPM, see [Flowcharts in Business process modeler (BPM)](https://docs.microsoft.com/en-us/...).
As part of the requirements for LCS solutions for localization & translation, localization ISV solution providers must undertake their regulatory watch by taking advantage of localization tools in Microsoft Dynamics Lifecycle Services (LCS).

### Set up an alerting project in LCS

Localization independent software vendor (ISV) solution providers must create a new project in Microsoft Dynamics Lifecycle Services (LCS) to record regulatory alerts.

Follow these steps to set up the project.

1. Add a new project by clicking the plus sign (+).

2. Enter a name that uses the following project naming convention: **REG-Alerts-Country/region name**

3. Enter a project description.

4. For the product name, specify the latest version of your Dynamics 365 Finance and Operations app.

5. For the product version, specify the latest version.

6. Specify the industry:
   - Select **Other** if the solution is related to all industries.
   - Select an appropriate specific industry.

7. For the methodology, specify **Sure step**.

8. Click **Create**.

### Invite participants to the project

Invite participants that should have access to the project, so that they can submit and review regulatory alerts. For information about how to invite users, see Configure Lifecycle Services (LCS) security.

### Access the regulatory alert submission service

1. In your LCS project, scroll to the right side of the page, and then, under **More tools**, click Localization and translation.
2. Select Dynamics Regulatory Alert Submission, and then click Enter. The Dynamics Regulatory Alert Submission page opens. Use this page to view any previous alerts that have been submitted by you or your organization.

Submit a regulatory alert

To enter a new regulatory alert, click the plus sign (+) at the top of the form, above the filter. The Submit regulatory alert wizard starts. Follow these steps to complete the wizard.

1. On the Search for existing items page, follow these steps:
   a. Use Issue search to identify whether a regulatory feature that is related to the alert already exists. Enter search terms to specify criteria such as a keyword, country/region, Microsoft Knowledge Base (KB) number, or Application Object Tree (AOT) object. For example, you can search for the payment term “SEPA” to return related items.
   b. After you've finished entering the search terms, click the Search button. Search returns all items that meet search criteria as it searches across both product issues and regulatory features.
   c. You can narrow down the search results by using the filter/criteria that are available.
   d. If you don't find the regulatory feature that you're searching for, you can submit a regulatory alert by clicking Submit regulatory alert on the bottom ribbon.

2. On the Attach business processes page, follow these steps:
   a. In the Global business process libraries field, you can select business process libraries.
   b. You can enter search criteria to return business processes that are related to a search term. The tool highlights these business processes in yellow.
   c. You can drag business processes into the marked area on the left side of the page. You can select one or many related business processes in the process list. After you've dragged business processes to the marked area on the left, you can edit them further. Deselect business processes by clicking the x.
   d. When you've finished, click Continue to go to the next page in the wizard. When you receive a
If no relevant business processes are found, you can skip this page in the wizard.

You receive a message that asks whether you want to add the selected business processes to the alert. You can click either Yes or Cancel.

3. On the **Describe the alert** page, follow these steps:

   a. Enter information for the alert in the appropriate fields. Required fields are indicated by a red asterisk (*). The following table provides more information about the submission fields.

<table>
<thead>
<tr>
<th>FIELD NAME</th>
<th>FIELD TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Text</td>
<td>A descriptive title to identify the area of impact. For example, you might enter <em>Changes in invoice document per Jan. 1st 2014</em>.</td>
</tr>
<tr>
<td>Description</td>
<td>Text</td>
<td>A brief, pragmatic overview of the law. The description should focus on issues that are relevant to enterprise resource planning (ERP). It should have three to ten lines of details, so that users gain a high-level understanding of the requirement without having to read the legislation first.</td>
</tr>
<tr>
<td>Country</td>
<td>Valid values list</td>
<td>The country/region that the legislation applies to.</td>
</tr>
<tr>
<td>Industry</td>
<td>Valid values list</td>
<td>A list of industries, if the requirement applies only to selected industries (for example, Public sector, Retail, or Communication).</td>
</tr>
<tr>
<td>Link to legislation</td>
<td>Text (URL format)</td>
<td>Add links to the published law, interpretation guideline, implementation guidance, or any other documentation that will be useful for understanding and implementing the requirement.</td>
</tr>
<tr>
<td>Feature reference</td>
<td>Text</td>
<td>The feature reference ID (if it's known).</td>
</tr>
<tr>
<td>Law enforcement date</td>
<td>Date</td>
<td>Date from which impacted customers must comply with the law</td>
</tr>
<tr>
<td>Government announcement date</td>
<td>Date</td>
<td>The date when the authority announced the change.</td>
</tr>
<tr>
<td>FIELD NAME</td>
<td>FIELD TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Latest filing date</td>
<td>Date</td>
<td>The deadline for the first submission of the new/changed report.</td>
</tr>
<tr>
<td>Company name</td>
<td>Text</td>
<td>The company name for the person who is submitting the alert.</td>
</tr>
<tr>
<td>Contact name</td>
<td>Text</td>
<td>The contact name of the person who is submitting the alert.</td>
</tr>
<tr>
<td>Contact email</td>
<td>Email address</td>
<td>The contact email address of the person who is submitting the alert. This value must be in a valid email address format.</td>
</tr>
<tr>
<td>Business processes</td>
<td>Default</td>
<td>The business processes are selected through the Submit regulatory alert wizard and entered automatically on the submission page.</td>
</tr>
<tr>
<td>Comments</td>
<td>Text</td>
<td>Enter any additional information that is related to the alert, and that might be useful for understanding or implementing the requirement.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>You can add multiple comments. Click Submit to save comments separately to the submission page. Comments are saved in order of date.</td>
</tr>
<tr>
<td>Attachments</td>
<td>Upload</td>
<td>Add alert attachments by using the attachment tool. Click the Upload button to open File explorer. After you select and upload a file, the file is appears as a linked file. You can upload up to three files, each of which can be up 5 MB. To delete a file that you've uploaded by mistake, click Remove under the file title. <strong>Important</strong>: Attachments must be publicly available materials. They can't be propriety or customer/partner specific.</td>
</tr>
<tr>
<td>Consent check box</td>
<td>Check box</td>
<td>Give appropriate consent to being contacted. The Submit button doesn't become available until you select this check box.</td>
</tr>
</tbody>
</table>

b. When you've finished adding all the required fields, you must provide appropriate consent by selecting the consent check box: By creating this alert I allow Microsoft to contact me in the future for any further information related to this alert. Read the Microsoft Dynamics Lifecycle Services privacy statement for more information.

c. After you select the check box, the Submit button becomes available. Click Submit to submit the alert. If you've partially completed an alert, you can save the information that you've already entered for later completion or review. Click Save before you submit the alert.
d. When you receive a confirmation message that states that the alert has been successfully submitted, click **Done** to exit the wizard. If you chose to save the alert before you submitted it, an alert ID is generated, and you are notified that the alert has been saved.

Review regulatory alerts that have been entered into the project

To view the regulatory alerts that have been entered into your alerting project, use the alert grid. This grid provides a high-level list view of the alerts that have been submitted, and shows the alert title, country/region, law enforced date, and so on.

You can search the contents of the grid by using the filter/search field and then selecting from the default search options. You can drill into the detail of an alert by clicking the alert ID, which is a hyperlink. The completed alert submission page opens, where you can review the alert details, and also any comments and attachments.

Process submitted alerts options

After an alert has been submitted to the LCS alert project, you can process it from the grid view by clicking the alert ID hyperlink. Project owners can then change the status of the alert to notify project members whether action will continue to be taken for the alert. Options appear when you drill into details of the alert.
The following table describes the processing options that are available for alerts.

<table>
<thead>
<tr>
<th>STATUS</th>
<th>STATE</th>
<th>ACTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submitted</td>
<td>The alert has been submitted to (entered into) LCS by a Loc. Community member.</td>
<td>The alert can be either received or rejected.</td>
</tr>
<tr>
<td>Received</td>
<td>The alert has been reviewed by an internal Sol. Partner resource and has been received for further review.</td>
<td>The alert is marked <strong>Received</strong>. A comment and feature reference can be added.</td>
</tr>
<tr>
<td>Rejected</td>
<td>The alert has been reviewed by an internal Sol. Partner resource and has been rejected.</td>
<td>The alert is marked <strong>Rejected</strong>. A comment must be added to explain why the alert was rejected.</td>
</tr>
<tr>
<td>Reopened</td>
<td>The alert was rejected but has been reopened by a Loc. Community member, and new/additional information has been entered (a comment is mandatory).</td>
<td>The alert can be reviewed again to assess whether it should be received or rejected.</td>
</tr>
</tbody>
</table>

**NOTE**

Submitted alerts can be rejected for various reasons. Here are some examples:

- The alert is too vague to identify the underlying localization feature.
- The alert is related to an area where no features are localized.
- The alert is related to an area that isn't currently supported by Finance and Operations functionality.

Alerts can be stored in LCS as references.

Further processing, such as potential engineering of a related feature, will be handled in the ISV solution provider’s existing systems.
As part of the requirements for LCS solutions for localization and translation, if a localization solution for previous versions of Dynamics 365 Finance and Operations apps contains both regulatory and competitive features, localization ISV solution providers must split the solution into separate models for each feature type. This article provides information about this requirement.

Multinational customers must comply with regulatory requirements in all countries/regions where they deploy Finance and Operations apps. At the same time, these customers want to minimize the cost of code maintenance. Therefore, if a localization solution for previous versions contains both regulatory and competitive features, the solution must be split into separate models, so that customers can adopt and deploy the features that they require. An effort has been made to split the Application foundation and Application suite stack into multiple models.

The number of models is expected to grow over time. Splitting a monolithic code base provides many benefits, such as better scalability, manageability, and serviceability. The localization requirement to split a localization solution into more granular models builds on this effort. The goal is to provide the same benefits to multinational customers.

After you’ve classified features as either regulatory or competitive, as described in Classification of localization features, split the code for these features into at least two models, one model for the regulatory features and at least one model for the competitive features. If the competitive features can be split further (for example, into features that are related to Commerce and features that are related to Fixed assets), it's a good idea to split them. However, further splitting isn't mandatory. For more information about how to split the stack into multiple models, see Model split.
This topic describes how to use Microsoft Dynamics Lifecycle Services (LCS) to submit alerts through the Dynamics regulatory alert submission service. This topic also explains how to track planned and released regulatory features through LCS Issue search.

Accessing the regulatory alert submission service

In Dynamics Lifecycle Services (LCS), in your project, scroll to the right side of the page, and then, under More tools, click the Alert service tile.

The Dynamics regulatory alert submission page appears. You can use this page to view any alerts that have previously been submitted by you or your organization.

Submitting a regulatory alert

To enter a new regulatory alert, click the plus sign (+) at the top of the Dynamics regulatory alert submission page, above the filter. The Alert submission wizard starts. You can complete the following tasks in this wizard:

- Search for existing regulatory items.
- Attach business processes.
- Describe an alert.
- Confirm submissions.

Search for existing regulatory items

Use Issue search to identify whether a regulatory feature, that is related to the alert, already exists.

1. Enter a search term, such as a keyword, country/region, Microsoft Knowledge Base (KB) number, or Application Object Tree (AOT) object. Click the search button. Any items that include the search term, in either product issues or regulatory features, appear in the search results. You can narrow the search by using the filters that are available.

2. If you don’t find the regulatory feature that you’re looking for, you can submit a regulatory alert by clicking Submit regulatory alert at the bottom of the browser window.

Attach business processes

1. In the Global business process libraries list, select business process libraries.

2. Enter search criteria to find business processes that are related to the search term. These business processes are highlighted in yellow.

3. In the list on the right side of the page, select one or more related business processes, and drag them into the field on the left. After you’ve finished, you can edit them further by clearing the selection of business processes.

4. Add the selected business processes to the alert.

Describe the alert

1. Enter information about the alert in the appropriate fields. Required fields are indicated by a red asterisk.

The following table provides more information about the fields on the Describe the alert page.
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Enter a descriptive title to identify the area of impact. For example, enter <em>Changes in invoice document as of January 1, 2018</em>.</td>
</tr>
<tr>
<td>Description</td>
<td>Enter a brief overview of the law. Your description should focus on issues that are relevant to enterprise resource planning (ERP), so that users can understand the requirements at a high level without having to read the legislation first.</td>
</tr>
<tr>
<td>Country</td>
<td>Select the country or region that the legislation applies to.</td>
</tr>
<tr>
<td>Industry</td>
<td>Select the industry, if the requirement applies only to specific industries. For example, select <em>Public sector, Commerce, or Manufacturing</em>.</td>
</tr>
<tr>
<td>Feature reference</td>
<td>Enter the feature reference, if you know it.</td>
</tr>
<tr>
<td>Law enforcement date</td>
<td>Select the date when affected customers must start to comply with the law.</td>
</tr>
<tr>
<td>Government announcement date</td>
<td>Select the date when the authority announced the change.</td>
</tr>
<tr>
<td>Latest filing date</td>
<td>Select the deadline for the first submission of the new or changed report.</td>
</tr>
<tr>
<td>Link to legislation</td>
<td>Enter one or more links to the published law, interpretation guideline, implementation guidance, or any other useful documentation that will help users understand or implement the requirement.</td>
</tr>
<tr>
<td>Company name</td>
<td>Enter the company name for the person who is submitting the alert.</td>
</tr>
<tr>
<td>Contact name</td>
<td>Enter the name of the person who is submitting the alert.</td>
</tr>
<tr>
<td>Contact email</td>
<td>The email address of the person who is submitting the alert.</td>
</tr>
</tbody>
</table>
2. After you’ve finished entering all the information, select the consent check box (By submitting this regulatory alert, I consent to Microsoft contacting me for additional information about this alert. Microsoft Privacy Statement.). When you select the check box, the Submit button becomes available.

3. Click Submit to save and submit the alert.

If you don’t have all of the required information, or if you’re not yet ready to submit the alert, you can save a partially completed alert.

Confirm your submission
- When the alert is successfully submitted, you receive a confirmation message. Click Done to exit the wizard.
- If you save the alert before you submit it, an alert ID is generated, and you receive confirmation that the alert has been saved.

Track the status of regulatory features in Issue search
You can use Issue search in LCS to find planned and released regulatory features, and any associated localization documentation, certifications, and reports. To narrow your search to regulatory features, use the following filters:

- **Category** - Select Regulatory feature only.
- **Country/region** - Click > to select the country/region that you’re interested in.

To narrow the search even more, you can apply the following additional filters

- **Product** - Select the products and product versions that you’re interested in.
- **Status** - Select specific statuses.
This topic explains how to get help with Finance and Operations apps or Microsoft Dynamics Lifecycle Services (LCS).

<table>
<thead>
<tr>
<th>TASK</th>
<th>MORE INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ask the community.</td>
<td>Go to the Dynamics 365 Community page to get help with your questions from the Microsoft Dynamics community.</td>
</tr>
<tr>
<td>Get help with questions about licensing.</td>
<td>Contact your partner or a Microsoft sales representative.</td>
</tr>
<tr>
<td>Use the <strong>Issue search</strong> tool.</td>
<td>In LCS, use the <strong>Issue search</strong> tool to quickly search for Microsoft Knowledge Base (KB) articles, hotfixes, and workarounds for reported issues. You can see which reported issues are in the process of being fixed for a specific functional area, and which issues have already been fixed. For more information, see <strong>Issue search (Lifecycle Services, LCS)</strong>.</td>
</tr>
<tr>
<td>Get support within your Finance and Operations app.</td>
<td>Select the <strong>Help</strong> button (?) in the upper-right corner of the app, and then select <strong>Support</strong>. Issues are reported on the <strong>Active issues</strong> tab in LCS. There, admins can determine whether they should provide in-house support or submit the issues to Microsoft.</td>
</tr>
</tbody>
</table>
| Open a support ticket with the Microsoft Support team. | In LCS, the **Support** tile opens a tool that helps you manage support incidents. To submit issues directly to Microsoft, select the **Support** tile in your LCS project. You can then submit issues in two ways:  
  - On the **Active issue** tab, select your issue, and then select **Submit to Microsoft**.  
  - On the **Submitted to Microsoft** tab, select **Submit an incident**, and then follow the on-screen instructions to submit the incident. After you submit an incident, you will receive an email message from the Microsoft Support engineer who is assigned to your case. |
| Request new features and functionality.          | Visit Dynamics 365 Application Ideas to view, search, or vote for existing ideas, or to add new ideas. |
Finance and Operations apps provides many self-service support options and support through 21Vianet.

**Self-help resources**

- Finance and Operations application documentation
- Help resources for Supply Chain Management
- Finance and Operations apps - operated by 21Vianet in China
- Dynamics community
- Microsoft Learn

**Assisted support**

- Open a support request

**Presales support**

Pre-sales support phone number: +86 400-886-6134

Pre-sales support provides assistance on subscription features and benefits, plan comparisons, pricing and licensing, and helps to identify the right solution to meet your business needs. In addition, pre-sales support can help you find a Partner, and purchase and sign up for a trial. You can call during local business hours, Monday through Friday.

**Billing and subscription management support through 21Vianet**

Billing and subscription support telephone number: +86 400-089-0365.

Assistance for billing and subscription management issues is available online or by telephone Monday through Friday during local business hours 9:00 to 18:00 China Standard Time (CST). Billing and subscription management support can be accessed using the same phone number and online service request process as with technical support.

Here are some examples of billing and subscription management issues:

- Signing up for a trial or purchasing a subscription.
- Converting from a trial subscription to a paid subscription.
- Understanding the bill.
- Renewing a subscription.
- Adding or removing licenses.
- Canceling a paid subscription.

**Assisted Technical support through 21Vianet**

When you experience a technical issue with your deployment, report it to 21Vianet through the LCS portal or by calling the support number at +86 400-089-0365. Technical support hours of operations are Monday through Friday during local business hours 9:00 to 18:00 China Standard Time (CST).

A support request (SR) is handled within hours, depending on the severity of its impact to your business.
• **Critical business impact** - You will receive an initial response within 1 hour or less, and a support representative will work continuously, all day, until the problem is resolved. You will be expected to allocate appropriate resources to work on the request until the problem is resolved and provide accurate contact information to the support personnel handling your case.

• **Non-critical business impact** - You will receive an initial response within 8 hours or less. You will be expected to provide accurate contact information to the support personnel handling your case.

**Get Premier support**

If you run mission-critical solutions, Premier support offers additional value:

- Proven advisory services designed to maximize your Dynamics 365 investment.
- A designated service delivery manager committed to improving your Dynamics 365 experience.
- Top priority reactive support to help ensure service continuity.

For details about purchasing Premier support, contact your Microsoft Account team. If you have a Premier support plan you can contact support via My Premier Online.

**Additional resources**

- Dynamics 365 support site for 21Vianet (Chinese)
- Finance and Operations apps - operated by 21Vianet in China
- Model-driven apps in Dynamics 365 - operated by 21Vianet in China
- Dynamics 365 Privacy statement (Dynamics 365 隐私声明)
- Dynamics 365 Service Level agreement (世纪互联在线服务的服务级别协议)
- Dynamics 365 Legal information (Dynamics 365 法律信息)
- Service terms for Dynamics 365 Lifecycle Services
- OSPT of Dynamics 365 (世纪互联在线服务的服务级别协议)
- Azure Docs (in Chinese)
- Azure China 21Vianet
Prerequisites

Before you can set up technical support, you must acquire a Microsoft Azure Active Directory (Azure AD) account. This account is created when you set up a subscription for one of the Microsoft Dynamics 365 Finance and Operations apps.

Create an Azure DevOps project

The Support tile in a Lifecycle Services (LCS) project uses Azure DevOps to store issues that are submitted through the client and issues that are manually created from the Support tile in LCS. This functionality requires that an Azure DevOps project be configured in the LCS project that you want to use for support. All users who need to use the Support tile to submit an issue must have access to the Azure DevOps project, and must authorize LCS to access Azure DevOps on their own behalf. Most users don't have access to LCS or Azure DevOps. Therefore, in the Azure DevOps project, you should create a special system account that can be used to submit issues.

Create a new Azure DevOps project

2. Click Sign in in the upper-right corner.
3. Sign in by using an AAD account that is in the tenant that your subscription is linked to. If the browser already has your credentials, you won't see the sign-in page and should instead click your name in the upper-right corner.
4. On the right side of the page, under Accounts, click Create a free account now.
5. Specify an account URL, and then click Create Account.
6. Name your project, and specify a process template. Your project should now be created.

Add users to the Azure DevOps project

1. In the upper-left corner, click Team Services.
2. On the Users tab, click Add, and invite users who will use the Support experience to the Azure DevOps account. For each user that you invite, select either Basic or Stakeholder.
3. In the upper-left corner, click Team Services.
4. Click Browse, and browse to the project that you created in the previous procedure.
5. In the Members section of the project home page, click Add, and add the users that you invited in step 2.

Create the Support system user

1. Create a new user in your Azure AD tenant, and enter a descriptive name, such as LcsCpsSystemAccount.
2. In the upper-left corner, click Team Services.
3. On the Users tab, click Add, and invite the system user that you created in step 1. For this user, select Stakeholder.
4. In the upper-left corner, click Team Services again.
5. Click Browse, and browse to the project that you created earlier.
6. In the Members section of the project home page, click Add, and add the system user.
Retrieve the personal access token for the Support system user

1. Sign out of Team Services by clicking the user name in the upper-right corner and then clicking Sign out.
2. Sign in to Team Services by using the Support system account that you created in the previous procedure.
3. In the upper-right corner, click the user name, and then click My profile.
4. On the Security tab, on the Personal access tokens tab, click Add.
5. Enter a description, such as LCS Support system account.
6. Select an expiration date of one year.
7. Click Selected scopes, and then select Work items (read and write).
8. Click Create token.
9. Copy the token and paste it in a safe location because it won't be accessible after you move away from the page.

Configure LCS

1. Sign in to LCS by using an account that has the Owner role for the LCS project that the application is deployed in.
2. Open the project in LCS.
3. Click Project settings, and then click the Azure DevOps link.
4. Click Setup Azure DevOps.
5. In the Azure DevOps site URL field, enter the URL of the Azure DevOps project that you created in the previous section.
6. In the Personal access token field, enter the personal access token that you created in the previous section.
Create an issue

The Support experience has been updated to show updates that are published by Microsoft. In the client, on the top bar, click ?, and then click Support.

WARNING

If you have an on-premises deployment, the option to search for existing issues and submit a support incident from the on-premises client to your Azure DevOps project is not available.
NOTE
If you haven't already connected to Lifecycle Services (LCS), a dialog box will display where you can connect. Click the link to connect before proceeding.

**Connect to Lifecycle Services**

Authorization must be given to Dynamics 365 for Operations in order to access Lifecycle Services. Click the link below to begin the authorization process. Once authorization has been granted return to this page and click the Ok button.

*Click here to connect to Lifecycle Services*

Search for a fix

After you connect to LCS, you can search for existing Microsoft published updates and fixes. Enter your issue in the **Search** box and press **Enter**.

NOTE
If you don't want the functionality to search for existing fixes enabled for all users, you can remove the **SearchExistingFixes** duty from the System user role and add it to only those roles which you want to have this functionality. Search results are based on the Microsoft Issue Search data that is relevant to your environment. Fixes that you have already installed will not be included in your search results. To view a specific result, click the link to view the details.

Based on the duties assigned to you, you will see either the **Download view** or the **Request view**.

- **Download view** - By default, this view is only available to system administrators. From this view, you can directly download the hotfix.

  NOTE
  The duty **DownloadHotfix** controls the ability to directly download fixes from LCS rather than requesting them. Only system administrators will have access to it by default. If you want to assign this duty to users other than system administrators, you can do so by adding the duty to the selected roles.

- **Request view** - By default, this view is available to all users who are not system administrators. From this view, you can make a request to download the hotfix. After you submit your request to download the hotfix, a work item will be created in the Azure DevOps project that is associated to your LCS project. The customer IT admin can view all requested hotfixes by clicking the **Support** tile in LCS and then clicking the **Hotfix requests** tab.

**Search for project work items in Azure DevOps**

The Azure DevOps administrator can publish project work items to your organization users by tagging the work items with **#SearchableInFinanceAndOperations**. The tagged work items will be searchable for users from the client support search box. The search result will include tagged Azure DevOps work items in addition to Microsoft published updates and fixes. The following graphic shows a tagged Azure DevOps work item for publishing.
When you search for published Azure DevOps work items using the support search box, search results show the work item’s type, title, state, and description in a new browser tab with view mode. Users with proper permissions can edit the work item in Azure DevOps. The following graphic shows the search result of a published Azure DevOps work item.

The published Azure DevOps work items are only visible to your organization's users.

Create and submit a new issue
If you don’t see a fix in the search results, you can create a new issue by clicking Create. This is the same functionality that is available for previous releases and is documented in earlier procedures.

Work with issues in LCS

View issues
In the LCS Support tile, issues are stored as work items in the Azure DevOps project that is associated with the LCS project. Specifically, issues are stored as work items of the Issue or Impediment type, depending on the type of Azure DevOps project, in the AxAndLcsGeneratedIssues area. Every work item of one of those types in that area will be included in the list of issues in the Support tile. If an issue is modified in Azure DevOps, the changes will be reflected in Support issues. Issues can be assigned to any user in the Azure DevOps project. Users don't need to have access to LCS to work with issues in Azure DevOps.

1. Go to lcs.dynamics.com, and sign in.
2. Open the LCS project that is associated with the environment that you want to view issues for.

3. Click the Support tile. A list of the issues that have been created appears.

### Edit issues

1. In the Issues grid, click the title of an issue.

2. If necessary, sign in to Azure DevOps by using an account that has access to the Azure DevOps project that you set up in the first section of this topic, Create an Azure DevOps project.

   **NOTE**
   
   There is an issue in Azure DevOps, where the link to edit work items doesn’t work correctly if sign-in is required. If you see the Assigned to me query after you sign in to Azure DevOps, go back to LCS, and click the title of the issue in the issue grid again.

3. The Azure DevOps editor opens. Edit the issue, and then save your changes. The changes will be reflected in the Support tile.

### Submit an issue to Microsoft

You can submit issues to Microsoft support. When you submit an issue to Microsoft, the information and attachments in the issue can be included in the Microsoft support incident.

LCS users must have a valid Microsoft support plan to submit issues to Microsoft. If you have trouble submitting issues to Microsoft, work with your administrator to make sure that your LCS credentials are added to or associated with your organization’s support plan with Microsoft Partner Source Business Center.

1. In the Issue grid, select the issue to submit to Microsoft, and then click Submit to Microsoft.

2. If your account is associated with multiple support organizations, select the organization to use to create the Microsoft support incident.

3. Use Issue search to verify that your issue hasn’t already been solved.

4. If Issue search doesn’t provide a solution to your issue, click Create incident at the bottom of the page.

5. Share diagnostic data with the Microsoft support team. By providing version information, your issues can be resolved more quickly.

6. Describe your issue, provide your contact information, and then click Submit.

### Support settings

**NOTE**

The information in this section is not applicable to on-premises deployments.

When you deploy your application from Lifecycle Services, no configuration is required, because the Support tool automatically saves any issues to the same LCS project that Finance and Operations was deployed from. To verify the LCS project that Support uses, go to System administration > Setup > System parameters, and
then click Help > Support Contact.

Prevent users from creating issues from the client

By default, the System user role has the privilege, SysLCSCPSIssueEntry assigned. This privilege controls access to the Contact your support team menu item on the Help menu. If you want to prevent users from being able to create and submit issues from the client, remove this privilege from the System user role.
Open a new incident

1. In LCS, go to the project for which you want to file a support incident.

2. Click the Support tile.

3. On the Submitted to Microsoft tab, click the Submit an incident button.

4. Select an issue category.

5. Select an issue area.

6. In the Describe your issue area, enter the following:
   - Select Yes if the issue occurred in an environment. Select the environment name.
   - Enter a short description of your issue in the Title field.
   - Provide details about the issue detail and the steps needed to reproduce the error.
   - If applicable, enter an error message.
   - If possible, attach screenshots that illustrate the problem. To do this, click Attach file from
7. Enter the primary contact information. These contact details will be used by the customer support team to contact you about the case.

8. Select the support contract and the severity level.

- Support contracts for on-premises environments have a limited incident count.
- Support contracts for cloud environments have an unlimited incident count.
- For on-premises products or cloud environments, from the list of available support contracts, select the support option to use if you have multiple tier support contracts.

9. Click **Submit**.

After you click **Submit**, an incident is created and added to the **Incidents** list. You will receive an email message from the Microsoft Support Engineer assigned to your case.

**Support plans in Lifecycle Services**

Support plan entitlements are derived based on several different identifiers. Not all will apply to your situation. If you are missing a support plan or entitlement in LCS, determine which identifier is needed to tie it to your project in LCS. If there is more than one organization, note which one is current by clicking on your name in the upper-right corner of LCS. Select the organization that applies to your scenario and contains the benefits that you want to utilize.

**Unique contract ID/access ID**

The following online support plans require a unique contract ID/access ID combination linked to your sign-in in LCS:

- Unified
- Premier
- Advanced support for partners

If you do not know your unique contract ID/access ID combination, contact your Microsoft account manager to have an ID created for you.

To link your contract ID/access ID to your account, complete the following steps:

1. From within a project, select **Support** from the main menu, and then select **Manage Support plans**.

2. Select **Add contract**.

3. Enter your access ID and your password or contract ID, and then select **Add contract**.
**PartnerSource Business Center account**

The following support plan incidents can be used as part of your PartnerSource Business Center (PSBC) account if they exist:

- Advanced support for partners on-premises incidents.
- Advantage or Advantage + on-premises incidents.
- Other pay per incident types of plans with an existing incident count in PSBC.

If you do not find the PartnerSource Business Center account, ensure that your sign in is added as a professional in your organization in PSBC. Make sure that you are signing in with the same Microsoft or work account login. This account is only applicable in an on-premises project.

**Sign-in specific options**

The following incidents and support benefits will appear based on your sign in, if applicable:

- MPN gold and silver incidents.
- Signature cloud support.
- Individual incidents and 5 packs purchased on [support.microsoft.com/supportforbusiness].

**NOTE**

Incidents must be purchased with a Microsoft account such as @hotmail.com or @outlook.com. Work or Azure Active Directory accounts cannot have incidents tied to them.

**Tenant subscription**

The following entitlements will appear based on your subscription and ProDirect purchases within your tenant organization:

- Subscription
- ProDirect

**Software assurance**

The following entitlements can be added by linking a subscription number and contact email:

- Software assurance
To add, select **Add a Software Assurance plan** when you create the support incident. Enter the subscription number and the contact email, and then click **Continue**.

Report production outage

For a quick and effective way to escalate issues to Microsoft Support in the event that the services in a production environment are degraded or become unavailable, see **Report a production outage**.

Phone support

We prefer that you contact Support following the steps in **Open a new incident**. If you’re unable to open a new incident in LCS, phone support is available using **Premier phone support**.
To ensure a good experience during the implementation of a project and after go-live, it's important that you understand the different types of servicing that are available and how you can get the correct support for every scenario. This topic explains how to engage each type of support and learn about some of the tools available.

Microsoft tools and support help ensure the stability and effectiveness of your environment by providing infrastructure and application support. However, this support can be effective only if partners and clients correctly develop, test, configure, manage, and monitor the implemented system and its environments.

Supporting actions can be grouped into three categories:

- **Self-service**
- **Service request**
- **Support request**

Depending on the category, the actions are triggered in different ways and might involve different lead times.

**Self-service**

Self-service actions can be triggered by users at any time and involve no lead time. Here are some of the reasons why a user might initiate a self-service action:

- Promote code (in non-production environments).
- Move databases between non-production environments.
- Upgrade a production environment.
- Turn maintenance mode on or off
- Pause upcoming automatic deployment of a service update.
- Restart services in non-production environments.
- Complete performance actions.

**Service request**

Service requests are usually triggered by creating a support ticket. They involve the cooperation of the Dynamics Service Engineering (DSE) team. Each request will involve designated lead time. A service request might be created to request environment deployment, for example.
Recommended practices for working with the DSE team

- Consider that there are turnaround times or Service Level Agreements (SLAs) for each type of service request.
- Don’t use a service request if a support request is more suitable for your case.
- To ensure that you make the correct type of request, see Service request types and SLAs.

Support request

Other scenarios can usually be resolved by opening a support request. These requests involve lead time. Here are some of the reasons for creating a support request:

- Do a point-in-time restore of a production environment after go-live.
- Flag a regression in a service update, and ask for an exception opt-out.
- Make performance-related requests (for tasks that can’t be completed through self-service).
- Activate a flying feature (for example, customer and vendor master data sharing).
- Resize a production environment. (You must first update and upload a new usage profile in the subscription estimator in Microsoft Dynamics Lifecycle Services [LCS].)
- Create an additional LCS project in the same tenant.
- Move the tenant of production environments.

Requesting support

An effective support process requires that a clear escalation path be defined. Project teams (client or partner) should be able to monitor and read environment telemetry, and they should be capable of doing any initial troubleshooting that is required.

A well-identified issue and a well-defined resolution process can make a difference in the effectiveness of the outcome.

An effective support request should include the following details:

- The environment where the issue occurs
- The process that the issue was identified in
- Reproducible steps to show where the issue occurred
- The expected result and the actual result
- Pre-investigation of the issue
- Additional elements (for example, the error message, screenshots, the session ID, and the trace)

High-severity support requests

A high-severity support request should include the following information:

- **Business impact:** How does the issue affect your business activities?
- **Financial impact:** How does the issue affect your business at a financial level?

A thorough analysis of each issue should be done before the issue is reported. The incident resolution will be much quicker if the support request includes detailed information, and if all available tools and telemetries have been used. By including all the correct information when you submit a support request, you reduce the amount of back-and-forth communication that is required to identify and replicate the issue. Therefore, you can save lots of time.

There are several support plans to choose from. Therefore, every business should be able to find a plan that meets its needs.

Before you request support, it’s important that you choose the correct level of severity. For information about how to identify the correct severity level and estimate the initial response time for your request, see Support.
LCS has an integrated set of tools that you can use to monitor LCS projects.

**Service health dashboard**

The Service health dashboard provides the health status for Office 365 services.

You can also view the service health through the Microsoft 365 admin center. Go to Health > Service health, or select the Service health card on the Home dashboard.

By default, the All services tab is selected on the Service health page. It shows all services and their current health state. A symbol and a value in the Status column indicate the state of each service.

If you're experiencing an issue with a Microsoft 365 service, but it isn't listed on the Service health page, you can notify Microsoft by selecting Report an issue and completing a short form.

The issue reporting will help to identify issues and how widespread they are. After incidents are identified, they will be shown on the dashboard under service health.

You can sign up to receive email communication. In this way, you can ensure that you're quickly alerted about issues that are identified in the tenant and their status change.

**Environment monitoring**

Environment monitoring is a set of tools that help you monitor and troubleshoot the health of your environments through LCS.

On a specific environment page for a project, the Monitoring section includes a Environment monitoring link that will take you to the Environment monitoring dashboard.

The following subsections describe some of the tools that are available.

**Overview**

The Overview section is common to most environment types. It provides a filterable way to trace user activity and to trace load by activity during a defined period.

**Activity**

The Activity tab lets you query raw logs. It provides predefined queries for the most common events and metrics to help you monitor your environment. Here are some examples of the predefined queries that are available:

- Slow queries
- Deadlocks
- Crashes
- Financial reporting issues
- Batch throttle
- Distinct user sessions

Additionally, you can add your own custom filters and export the logs to a comma-separated values (CSV) file for analysis.
Health Metrics

The **Health Metrics** dashboard provides a series of line charts that are filtered by instance (AOS or Batch AOS) and time frame. On the **AOS** tab, you can observe SQL execution. On the **System** tab, you can observe system memory and CPU utilization over time. This tool lets you easily identify behavioral changes. Therefore, it can help you trace issues over time and the impact of changes in the solution.

For additional content related to monitoring LCS environments, see [*Performance troubleshooting using tools in Lifecycle Services (LCS)*](#).

Although it’s important that you monitor your environments, you don’t have to be on constant lookout. Microsoft also uses the emails that are provided in the LCS notification list to alert you about important issues and actions that you must take, and to provide preventive guidance for the implementation itself.
This topic points to content for system administrators of Finance and Operations. This content will help you configure the system so that it works smoothly and effectively for your organization.

One Version

In July 2018 we announced a change to the way we deliver Dynamics 365 updates that will help you stay current in a consistent, predictable, and seamless manner. The following topics are intended to provide clarity on the Finance and Operations service updates, processes, and tools you can use to stay current.

- One Version service updates overview
- One Version service updates FAQ
- Service update availability
- Apply updates to cloud environments
- Configure service updates through Lifecycle Services (LCS)
- Pause service updates through Lifecycle Services (LCS)
- Get notified about service updates through Lifecycle Services (LCS)

Implementation management with Lifecycle Services

Microsoft Dynamics Lifecycle Services (LCS) is a collaboration portal that provides an environment and a set of regularly updated services that can help you manage the lifecycle of your Finance and Operations implementations.

The lifecycle of an implementation spans many phases from pre-sales through Analysis, Design and Development, Test, and Deployment to Operation, possibly in multiple iterative roll-outs. It can last a few months to multiple years, based on the scope and complexity of the project and the chosen deployment model, for example, in the managed cloud or on-premises.

The management of the implementation involves many different stakeholders from the customer and partner organizations and, especially in the cloud-hosted deployment model, from Microsoft. The implementation is supported through tools provided on LCS and through processes defined within the Microsoft FastTrack and through the partner’s implementation approach.

- Lifecycle Services resources
- Lifecycle Services (LCS) user guide

Deployment

You can deploy in the cloud or on-premises. Cloud deployments offer an enterprise resource planning (ERP) service that is fully managed by Microsoft. On-premises deployments are deployed locally in a customer’s data center.

- Software lifecycle policy and cloud releases
- Cloud deployment overview
- System requirements for cloud deployments
- On-premises deployment home page
- System requirements for on-premises deployments
Upgrade

An upgrade can involve moving to a new product version, migrating and upgrading code, moving to an update, or deploying a hotfix.

Although the processes for each type of upgrade are similar, they differ enough that you should review the topics for a specific task before you begin.

- Upgrades, updates, and hotfixes resources

Database management

For information to help you move a database to new environment and restore a database to a specific point in time, see Database movement operations home page.

Security

Finance and Operation apps uses role-based security. Access is granted only to security roles, not to individual users. Users are assigned to roles. A user who is assigned to a security role has access to the set of privileges that is associated with that role. A user who isn't assigned to any role has no privileges.

Role-based security is aligned with the structure of the business. The security roles that a user is assigned to depend on the user's responsibilities in the organization, and their participation in business processes. The administrator grants access to the duties that users in a role perform, not to the program elements that users must use.

Because rules can be set up for automatic role assignment, the administrator doesn't have to be involved every time that a user's responsibilities change. After security roles and rules have been set up, business managers can control day-to-day user access, based on business data.

- Role-based security
- Security architecture
- Encryption in Finance and Operations apps

Batch processing

Many tasks can be run as part of batch jobs. For example, batch jobs can include tasks for printing reports, doing maintenance, or sending electronic documents. By using batch jobs, you can avoid slowing down your computer or the server during typical working hours.

- Batch processing overview
- Batch processing and batch servers

Optimization advisor

- Optimization advisor overview
- Optimization advisor (video)
- Create rules for Optimization advisor

Office integration

The integration with Microsoft Office provides a set of productive, collaborative, and integrated user experiences that take advantage of the Microsoft Office suite. This functionality can help your organization become more efficient and effective.

- Office integration overview
Mobile

The Finance and Operations mobile app enables your organization to make its business processes available on mobile devices. After you enable the mobile workspaces for your organization, users can sign in to the app and immediately begin to run business processes from their mobile devices.

- Mobile app home page
- Available mobile workspaces

Process Automation

The process automation framework allows administrators to view and create automated processes that will be scheduled with the batch server. The added layer of visibility of scheduled work is presented in a calendar view that can be extended for use in application areas to allow non-system administrator users to view work that impacts their area.

- Process Automation home page

General administration

- Demo data overview
- Cross-company data sharing
- Add links to your organization’s legal terms and privacy statement
- License codes and configuration keys report
- Maintenance mode
- Preconfigured system accounts
- Export business-to-business (B2B) users to Azure Active Directory
- Set the session inactivity timeout
- Build OData metadata cache when AOS starts
- Configure and manage database logging
This topic explains how administrators can add links to their organization's legal terms and privacy statement in the About pane of Microsoft Dynamics 365 Finance, Supply Chain Management, and Commerce.

Organizations often need to ensure that the links to their legal terms and privacy statement are readily available and visible to users in order to meet legal and compliance requirements. Administrators of an organization can follow these steps to have the links to their legal terms and privacy statement be available in the About pane (Settings > About).

Add links

1. Go to the System parameters page and click Legal and Privacy. On this page:
   a. Enter the link to a page that outlines the legal terms for your organization.
   b. Enter the link to a page that outlines the privacy statement for your organization.

   **NOTE**
   Make sure that you enter the full URL, starting with either https or http.

2. Click Save.

3. If you are using Commerce, go to the Distribution schedules page. On this page:
   a. Select the 1110 – Global configuration job.
   b. Click Run now.

   **NOTE**
   To verify that the job completed, go to the Download sessions page.

Validate links

**Validate the links in Finance, Supply Chain Management, and Commerce**

To validate that the links have been added, on the toolbar at the top of the page, click the Settings icon, and then click About. In the Links section of the pane, you should see two new links:

- Your organization’s Legal terms
- Your organization’s Privacy and Cookies

Click these links to validate that the appropriate pages open.
Validate the links in Modern Point of Sale (MPOS) and Cloud Point of Sale (CPOS)

To validate that the links have been added, go to the Settings page. In the About section, click the links to validate that the appropriate pages open.
This topic points you to a report that lists the license codes and configuration keys available in Finance and Operations.

When you purchase Finance and Operations, all functionality is included. By default, some features and functionality that you do not use may be enabled. The administrator should disable the features that are not needed by disabling license codes and configuration keys.

When a license code or configuration key is disabled, the associated module or feature is removed from the user interface. Large sets of functionality, such as modules, are controlled by license codes. Many license codes, in turn, enable configuration keys that allow you to enable and disable functionality at a more detailed level.

To view the report

The License codes and configuration keys report, included with the Technical reference reports, lists each configuration key that is available. The report also indicates the license code and menu items associated with each configuration key.
Cross-company data sharing

This topic provides information about cross-company data sharing. Cross-company sharing is a mechanism for sharing reference and group data among companies in a Finance and Operations deployment. This feature resembles the virtual companies feature in Microsoft Dynamics AX 2012.

What is this feature and how does it work?

Cross-company data sharing lets you replicate (share) reference and group data among companies. Data integrity is verified before replication occurs.

Here are some examples of cross-company data sharing and the basic logic:

- The same payment terms and payment day definitions are used across 15 legal entities.
- The same terms of delivery are used across seven legal entities in three countries/regions.
- Records created, updated, and deleted in any of the companies within the policy will be replicated immediately, across all the companies.
- Fields that are not selected for sharing are maintained in each company and will not trigger any replication.
- As part of enabling a policy, it is optional to copy any existing records.

Cross-company data sharing has the following limitations:

- It can't be used to share transactional data between companies.
- Only reference and group data can be shared, or tables that have specifically been enabled. For example, Data Sharing Type is set to Duplicate.
- It supports replication of fewer than one million total records per job. This total is calculated as the number of shared records × the number of shared companies. The limit is increased to two million records from the Platform Update for version 10.0.10.
- It supports replication for up to 100 companies per policy. The limit is increased to 300 companies from the Platform Update for version 10.0.10.
- Only one level of child relationships is exposed. To protect data consistency, replication doesn't occur if another level is required.
- Fields that reference Financial dimensions, for example Ledger or Default dimension, can't be shared across companies. Dimensions hold a loose foreign key reference to the backing dimension data, which can reference both company-specific and non-company specific data. Determining the appropriate action to be taken for each dimension value has inherent complexity and would require a change from the current implementation, which could dramatically impact performance.
- It can't be used with dual-write.

Policies

Data sharing is managed by defined policies that are saved in data packages. Templates that Microsoft has tested and supports are available as downloadable data packages on Microsoft Dynamics Lifecycle Services (LCS). Policies let you control the following aspects of data sharing:

- The fields that are replicated
- The entities that participate in the replication
- The companies that participates in the sharing

The same company and table can only be in one policy. It is possible to share the same table in more than policy.
This can happen when the limits of records or companies are reached, or to create policies for tables that need to be shared differently for different country/regions.

**NOTE**

Only required foreign key fields are selected by default. Optional foreign keys need to be selected manually to be included. The best practice is to add one or more tables when selecting a foreign key field, unless the table has already been added.

Policy templates that Microsoft has tested and supports are available as downloadable data packages on Lifecycle Services (LCS).

**IMPORTANT**

Although customers can modify the Microsoft data templates that are available from LCS, this scenario isn't supported.

**Conflict resolution**

Validation rules are run when a sharing policy is enabled. If inconsistencies are detected, the user who implements the system can choose which records from which company should win.

**Considerations for successful data sharing**

Several entities in the Microsoft data packages have references that you must consider when you enable the entities. Some data sharing policies can’t be enabled if references don’t match. Other policies can be enabled, but you should use the Find inconsistency checker tool to verify that your data is consistent. Here are some examples:

- The Production group sharing policy has a reference to a company’s chart of accounts. Therefore, all companies that are added to this sharing policy must use the same chart of accounts.
- If you want to enable entities that use number sequences, the number sequence types must be the same across all companies in a sharing policy for those entities.
- Setup options must be the same across the companies that are involved in the sharing policy. Examples of setup options include the setting that specifies whether tax is included by default.

**When should I use cross-company data sharing?**

Use cross-company data sharing for the following business scenarios:

- Sharing of simple reference and group data in a single deployment
- Sharing among companies that have very similar configurations
- Sharing scenarios that have been explicitly tested by Microsoft

Cross-company data sharing isn’t supported for the following scenarios:

- Franchising solutions, where thousands of records are shared across thousands of companies.
- Sharing of transactional records for reporting or management purposes, such as consolidations.
- Sharing across deployments.
- Complex scenarios, such as replication of subtype/supertype tables or tables that have date effectivity rules.
- Tables that do not have a unique index.

**Customer and vendor master data sharing**

Customer and vendor master data sharing allows you to share customer and vendor data across multiple companies. If you would like to be considered for this feature, complete the Data sharing application and contact
With the release of Platform update for version 10.0.12, customer and vendor master data sharing can be enabled using the **Customer and vendor master data sharing** feature in the **Feature management** module. There is no need to complete a survey first. It is important to consider limits in the number of records and companies stated above.

---

**NOTE**

Default dimensions set up against a customer or vendor cannot be shared across companies. When configuring the customer or vendor record for cross-company data sharing, the **DefaultDimension** field is disabled, and cannot be included in the data sharing policy.

---

Default dimensions hold a loose foreign key reference to the backing dimension data, which can reference both company-specific and non-company specific data. Determining the appropriate action to be taken for each dimension value has inherent complexity and would require a change from the current implementation, which could dramatically impact performance.

---

### Download a cross-company data sharing template from LCS

1. Sign in to LCS.
2. On the home page, click **Shared asset library**.
3. In the **Asset type** list, click **Data package**.
4. Click any of the available data package files to download them.

For details about how to use a template, see [Configure financial cross-company data sharing](#).

### Currently supported cross-company data sharing templates

<table>
<thead>
<tr>
<th>PACKAGE NAME ON LCS</th>
<th>DATA SHARING POLICIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Financial data sharing templates</td>
<td>• Bank parameters</td>
</tr>
<tr>
<td></td>
<td>• Ledger journal names</td>
</tr>
<tr>
<td></td>
<td>• Payment days</td>
</tr>
<tr>
<td></td>
<td>• Payment schedules</td>
</tr>
<tr>
<td></td>
<td>• Payment terms</td>
</tr>
<tr>
<td></td>
<td>• Tax exempt codes</td>
</tr>
</tbody>
</table>

---
<table>
<thead>
<tr>
<th>PACKAGE NAME ON LCS</th>
<th>DATA SHARING POLICIES</th>
</tr>
</thead>
</table>
| Supply chain data sharing templates | • Barcode parameters  
• Barcode setup  
• Buyer group  
• Charges group  
• Commission  
• Destination code  
• Non-conformance type  
• Order entry deadline group  
• Order origin code  
• Order pool  
• Production group  
• Production pool  
• Reason for delivery  
• Supplementary item group  
• Terms of delivery  
• Work time calendar |

**Additional resources**

Configure financial cross-company data sharing (Task guide)
This topic provides information about maintenance mode in Finance and Operations. When maintenance mode is turned on, it provides a safe way for system administrators to make system changes that might affect system functionality. For example, configuration keys can be enabled or disabled. While maintenance mode is on, only system administrators and users who have the Maintenance mode user role can sign in to the system. By default, maintenance mode is turned off. When maintenance mode is off, you can’t edit the License configuration page.

Turn maintenance mode on and off on sandbox and production environments through Lifecycle Services

You can now turn maintenance mode on and off directly through Lifecycle Services (LCS) on your sandbox and production environments. Refer to the following steps to do this:

1. Go to the environment details page and on the Maintain menu, click Enable Maintenance Mode.
2. In the slider, set Turn maintenance mode on for the environment and select Confirm.
3. A servicing operation will begin and your system will go into maintenance mode.
4. On completion, the environment state will be In Maintenance. At this point, only the system administrator will have access to the environment.
5. After you are done making system-wide changes, you can turn off maintenance mode by clicking Disable Maintenance Mode under the Maintain menu.
6. This will start a servicing operation that takes your environment out of maintenance mode. You can see the progress of the operation in the environment details page.
7. After this is complete, your environment goes back to the Deployed state. Now all users can sign in to the environment.
8. You can check the environment history page to see when the maintenance mode was turned on or turned off. To get to the environment history page, select History and Environment changes on the environment details page.

Turning maintenance mode on and off for your sandbox and production environment is very similar to a servicing operation. If turning maintenance mode on or off fails, you will see options such as Resume, Rollback, and Abort. You also have the option to download the logs to troubleshoot why the operation failed.

Turn maintenance mode on and off in DevTest/Demo environments hosted in Customer's subscription

1. Establish an RDP connection to the developer machine.
2. On the developer machine, sign in to SQL Server by using the credentials for the axdbadmin user from LCS. Then switch to the AXDB database, and run the following command.

   ```sql
   update SQLSYSTEMVARIABLES SET VALUE = 1 where PARM = 'CONFIGURATIONMODE'
   ```

3. Restart the World Wide Web Publishing Service to reset IIS.
4. After the service is restarted, the system will be in maintenance mode.
5. When you've completed your maintenance mode activities, repeat steps 2 and 3, but set the value to 0 in step 2.

**Turn maintenance mode on and off for VHD-based environments hosted by customers**

You can turn on maintenance mode locally by running the following command.

```bash
J:\AosService\PackagesLocalDirectory\Bin\Microsoft.Dynamics.AX.Deployment.Setup.exe --metadatadir J:\AosService\PackagesLocalDirectory --bindir J:\AosService\PackagesLocalDirectory\Bin --sqlserver . --sqldatabase axdb --sqluser axdbadmin --sqlpwd ********--setupmode maintenancemode --isinmaintenancemode true
```

After running the command, you will want to restart the **World Wide Web Publishing Service** to reset IIS. The system will then be in maintenance mode.

The following table describes the parameters that are used in this command.

<table>
<thead>
<tr>
<th>PARAMETER NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>--setupmode maintenancemode</td>
<td>Use this parameter to inform the setup tool that the system will be put into or taken out of maintenance mode.</td>
</tr>
<tr>
<td>--metadatadir</td>
<td>Use this parameter to specify the metadata directory. You should use the default packages directory.</td>
</tr>
<tr>
<td>--bindir</td>
<td>Use this parameter to specify the binaries directory. You should use the default packages directory.</td>
</tr>
<tr>
<td>--sqlserver</td>
<td>Use this parameter to specify the Microsoft SQL Server. For one-box environments, use a period (.).</td>
</tr>
<tr>
<td>--sqluser</td>
<td>Use this parameter to specify the SQL Server user. You should use AOSUser.</td>
</tr>
<tr>
<td>--sqlpwd</td>
<td>Use this parameter to specify the SQL Server password.</td>
</tr>
<tr>
<td>--isinmaintenancemode</td>
<td>Use this parameter to turn configuration mode on or off. Use <strong>true</strong> to turn it on and <strong>false</strong> to turn it off.</td>
</tr>
</tbody>
</table>

**Enable (or disable) configuration keys**

After the instance of Application Object Server (AOS) is restarted, the system will be in maintenance mode. You can then enable configuration keys, as shown in the following screenshot.
If you try to access the system while in maintenance mode, but you aren't a system administrator or a user who has the **Maintenance mode user** role, you may receive an error message. 

You can turn off maintenance mode by running the following command.

```
J:\AosService\PackagesLocalDirectory\Bin\Microsoft.Dynamics.AX.Deployment.Setup.exe --metadatadir J:\AosService\PackagesLocalDirectory --bindir J:\AosService\PackagesLocalDirectory\Bin --sqlserver . --sqldatabase axdb --sqluser axdbadmin --sqlpwd ********* --setupmode maintenancemode --isinmaintenancemode false
```

After running the command, you will want to restart the **World Wide Web Publishing Service** to reset IIS. The system will then be out of maintenance mode.
Pre-configured system accounts are included on deployed environments so that Microsoft can manage and operate the Finance and Operations service and provide specific features to customers. The following table provides information about each account, including the purpose and use case for the account.

### IMPORTANT
Do not delete these system accounts. Deleting these accounts will cause a disruption in key functionality provided by Microsoft.

<table>
<thead>
<tr>
<th>ACCOUNT DETAIL</th>
<th>PURPOSE/USE CASE OF THE ACCOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axrunner</td>
<td>This account is used to monitor the health of the environment and provide alerts when necessary.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: This account is deprecated with self-service environments and is no longer used.</td>
</tr>
<tr>
<td>FRServiceUser</td>
<td>This account is the Financial Reporting service user account, which is used by the Management Reporter application for integrations with Finance and Operations.</td>
</tr>
<tr>
<td>RetailServiceAccount</td>
<td>This account is used for Retail services to connect to the Finance and Operations environment.</td>
</tr>
<tr>
<td>SysHealthServiceUser or Axping (depending on the deployed product version)</td>
<td>This account is used to monitor the availability and health of the environment and provide alerts when necessary.</td>
</tr>
</tbody>
</table>
You can automatically export business-to-business (B2B) users to Azure Active Directory (Azure AD).

In the past, B2B users were exported manually to a .csv file. Then the Azure AD tenant administrator had to use this file to manually add the users to Azure AD using the Azure portal.

To enable the automatic export feature, a one-time setup and configuration process must be completed. When the process is completed, you can use the **Provision Azure AD B2B user** workflow task to automatically export B2B users to Azure AD.

The one-time set up and configuration means that you'll need to:

1. Set up a B2B invitation service application in Azure AD.

### Set up a B2B invitation service application in Azure AD

The tenant administrator of your Azure AD tenant will need to complete the following steps.

1. Log on to the [Azure portal](#) as the tenant administrator.
2. Click **Azure Active Directory** > **Properties**.
3. Copy the **Directory ID** (this is the tenant ID) and save it. You will need this later.
4. Click **App registrations** > **New application registration**.
5. Enter the following information, and then click **Create**.
   a. In the **Name** field, enter the name of the application. For example: **B2B admin application**.
   b. In the **Application type** field, select **Web app /API**.
   c. In the **Sign-on URL** field, enter the URL for Finance and Operations.
6. Click the **App registrations** tab, click the newly created application, copy the **Application ID**, and save it. You will need this later.
7. Click **All settings** > **Required permissions** > **Add**.
8. In the **Add API access** pane, do the following:
   a. Click the **Select an API** tab. Click **Microsoft Graph**, and then click **Select**.
   b. In the **Select permissions** tab, select the following **application permissions** and set them to **Yes**:
      - Invite guest users to the organization
      - Read and write directory data
      - Read and write all users’ full profiles
   c. Select the following **delegated permissions** and set them to **Yes**:
      - Invite guest users to the organization
      - Read and write directory data
      - Read and write all users’ full profiles
Configure the B2B invitation service settings

1. Sign in to Finance and Operations as administrator.


3. Select Enabled.

4. Verify that the Tenant ID is the same as the Directory ID (which you noted in step 3 of the previous procedure).

5. In the Client ID field, enter the Application ID (which you noted in step 6 of the previous procedure).

6. Enter the key Value, copied from the above procedure, into the Application Key field.

7. Save the settings.

Now you can start using the Provision Azure AD B2B user workflow task in your workflows to automatically export B2B users to Azure AD.
Data maintenance enables simple scheduling processes that you can run to find or correct data inconsistencies in your environment.

Incorrect data can adversely affect your day-to-day, monthly, and yearly operations. Inconsistencies and errors that come from incorrect data has the potential to halt major events like year-end activities and can even halt your daily revenue streams and affect your organization's decision-making capabilities.

The *Data Maintenance Portal* is a tool that lets system administrators schedule and run various actions that will have a direct effect on the data or the system. Some actions can be scheduled to continuously look for opportunities to fix issues, and others can be run on demand to enact some change on the system. Currently there are three basic types of actions: direct, scanning, and fixing.

### Types of actions

- **Direct actions** can be run on-demand only, and can run tasks directly. Microsoft Support may use direct actions, which could be as simple as clearing a cache without the need for downtime or as complicated as running a reference scanner to aid the support process.

- **Scanning actions** will search your data, a few times a day, looking for problems in the data. The problems found will be reported to Microsoft. There are a number of system actions that may not yet have an automated fix, but will provide valuable data to Microsoft to improve the health of your data. Microsoft may reach out to you regarding problems found through this method.

- **Fixing actions** runs on the same cadence as a scanning action, but when an opportunity is found, it will schedule a fix to the data. Fixing actions are meant to be data idempotent and may not fix all of the data on the first run. We recommend that a fixing action only fixes a subset of data each time it runs. Over time, the data will reach a clean state without exposing a significant load on the system. This type of action may help facilitate an in-place upgrade of your system.

### Control of actions

To access the Data Maintenance Portal, administrators can go to *System administration > Periodic tasks > Data maintenance*. On this page, administrators can see the list of actions that are available, and the latest status of each action. Important information about the action can be found in the right panel. If the action can be scheduled, there will be a button labeled *Schedule* available at the bottom of the page. All actions can be run on demand, prior to being run by the automated schedule, by selecting the *Run now* button. System actions, defined by Microsoft, cannot be disabled or enabled.

**NOTE**

The recurrence of data maintenance processes are handled by the process automation framework as background processes. There are two main types of background processes: one for scanning for opportunities and one for running tasks. For more information, see *Process automation framework development*. 
This topic provides an overview of the elements of role-based security in Finance and Operations.

In role-based security, access is not granted to individual users, only to security roles. Users are assigned to roles. A user who is assigned to a security role has access to the set of privileges that is associated with that role. A user who is not assigned to any role has no privileges.

In Finance and Operations apps, role-based security is aligned with the structure of the business. Users are assigned to security roles based on their responsibilities in the organization and their participation in business processes. The administrator grants access to the duties that users in a role perform, not to the program elements that users must use.

Because rules can be set up for automatic role assignment, the administrator does not have to be involved every time that a user’s responsibilities change. After security roles and rules have been set up, business managers can control day-to-day user access based on business data.

Overview of role-based security

This section provides an overview of the elements of role-based security. The security model is hierarchical, and each element in the hierarchy represents a different level of detail. Permissions represent access to individual securable objects, such as menu items and tables. Privileges are composed of permissions and represent access to tasks, such as canceling payments and processing deposits. Duties are composed of privileges and represent parts of a business process, such as maintaining bank transactions. Both duties and privileges can be assigned to roles to grant access to Finance and Operations.

The following illustration shows the elements of role-based security and their relationships.

Security roles

All users must be assigned to at least one security role in order to have access to Finance and Operations. The security roles that are assigned to a user determine the duties that the user can perform and the parts of the user interface that the user can view.

Administrators can apply data security policies to limit the data that the users in a role have access to. For example, a user in a role may have access to data only from a single organization. The administrator can also specify the level of access that the users in a role have to current, past, and future records. For example, users in a role can be assigned privileges that allow them to view records for all periods, but that allow them to modify records only for the current period.
By managing access through security roles, administrators save time because they do not have to manage access separately for each user. Security roles are defined one time for all organizations. In addition, users can be automatically assigned to roles based on business data. For example, the administrator can set up a rule that associates a Human resources position with a security role. Any time that users are assigned to that position, those users are automatically added to the appropriate security roles.

Security roles can be organized into a hierarchy. The role hierarchy allows the administrator to define a role based on another role. For example, the sales manager role could be defined as a parent role of the manager role and the salesperson role. A parent role automatically inherits the duties, privileges, and conditions that are assigned to its child roles. Therefore, a user who is assigned to the parent role can perform all of the tasks that users in the child roles can perform. A role can have one or more child roles or one or more parent roles.

By default, sample security roles are provided. All functionality is associated with at least one of the sample security roles. The administrator can assign users to the sample security roles, modify the sample security roles to fit the needs of the business, or create new security roles. By default, the sample roles are not arranged in a hierarchy.

**Duties**

Duties correspond to parts of a business process. The administrator assigns duties to security roles. A duty can be assigned to more than one role.

In the security model, duties contain privileges. For example, the **Maintain bank transactions** duty contains the **Generate deposit slips** and **Cancel payments** privileges. Although both duties and privileges can be assigned to security roles, we recommend that you use duties to grant access to Finance and Operations.

You can assign related duties to separate roles. These duties are said to be segregated. By segregating duties, you can better comply with regulatory requirements, such as those from Sarbanes-Oxley (SOX), International Financial Reporting Standards (IFRS), and the United States Food and Drug Administration (FDA). In addition, segregation of duties helps reduce the risk of fraud, and helps you detect errors or irregularities.

Default duties are provided. The administrator can modify the privileges that are associated with a duty, or create new duties.

**Privileges**

In the security model, a privilege specifies the level of access that is required to perform a job, solve a problem, or complete an assignment. Privileges can be assigned directly to roles, however we recommend that you only assign duties to roles. This is so that the privileges are first grouped together into a duty, which makes it easier to maintain.

A privilege contains permissions to individual application objects, such as user interface elements and tables. For example, the **Cancel payments** privilege contains permissions to the menu items, fields, and tables that are required to cancel payments.

By default, privileges are provided for all features in Finance and Operations. The administrator can modify the permissions that are associated with a privilege, or create new privileges.

**Permissions**

Each function, such as a form or a service, is accessed through an entry point. Menu items, web content items, and service operations are referred to collectively as entry points.

In the security model, permissions group the securable objects and access levels that are required to run a function. This includes any tables, fields, forms, or server side methods that are accessed through the entry point.
This topic provides an overview of the security architecture of Finance and Operations.

When you understand the security architecture, you can more easily customize security to fit the requirements of your business. The following diagram provides a high-level overview of the security architecture.

**Authentication**

By default, only authenticated users who have user rights can establish a connection.

Microsoft Azure Active Directory (AAD) is a primary identity provider. To access the system, users must be provisioned into a Finance and Operations instance and should have a valid AAD account in an authorized tenant.

**Authorization**

Authorization is the control of access to Finance and Operations applications. Security permissions are used to control access to individual elements of the program: menus, menu items, action and command buttons, reports, service operations, web URL menu items, web controls, and fields in the Finance and Operations client.

Individual security permissions are combined into privileges, and privileges are combined into duties. The administrator grants security roles access to the program by assigning duties and privileges to those roles.

Context-based security controls access to securable objects. When a privilege is associated with an entry point (such as a menu item or a service operation), a level of access, such as Read or Delete, is specified. The
authorization subsystem detects the access at run time, when that entry point is accessed, and applies the specified level of access to the securable object that the entry point leads to. This functionality helps to ensure that there is no over-permissioning, and the developer gets the access that was intended.

For more information, see Role-based security.

Data security

Authorization is used to grant access to elements of the program. By contrast, data security is used to deny access to tables, fields, and rows in the database.

Use the extensible data security framework to supplement role-based security by restricting access to table records based on security policies. A security permission, as part of a user role, increases the access a user has to data, while a security policy decreases access to data.

For more information, see Extensible data security policies.

 Additionally, the Table Permissions Framework helps protect some data. Data security for specific tables is enforced by Application Object Server (AOS).

Auditing

Auditing of user sign in and sign out is enabled, which means that the system logs when a user signs in or out of the application. A sign out is logged even if the user's session expires or ends.

A system administrator or security administrator can access the audit logs by going to the User log page (System administration > Inquiries > User log).
Encryption at rest

Microsoft uses encryption technology to protect customer data while at rest in an environment’s SQL Server database and Azure Storage.

All instances utilize Microsoft SQL Server Transparent Data Encryption (TDE) and Azure Storage encryption to perform real-time encryption of data when written to the disk at rest.

Finance and Operations apps use server-side encryption using service-managed keys. All key management aspects such as key issuance, rotation, and backup are handled by Microsoft.

In addition to the default encryption at rest provided above, you can use the encryption API available in the Global X++ class. The methods Global::editEncryptedField() and Global::editEncryptedStringField() use the environment-specific data encryption certificate to perform data encryption and decryption. You can use these methods as an additional layer of protection beyond the default encryption at rest technology used for data storage.

Encryption in transit

Connections established between customers and Microsoft datacenters are encrypted, and all public endpoints are secured using industry-standard Transport Layer Security (TLS) 1.2. TLS effectively establishes a security-enhanced browser-to-server connection to help ensure data confidentiality and integrity between desktops and datacenters.

Supported TLS versions

Finance and Operations apps support TLS 1.2 only. Earlier TLS versions, 1.0 and 1.1, are not supported.

Supported cipher suites

Finance and Operations apps only support the following cipher suites:

- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256

Additional resources

- Azure Data Encryption-at-Rest
- Microsoft SQL Server Transparent Data Encryption (TDE)
- Azure Storage encryption
- Insider tips on development
The session inactivity timeout setting represents the amount of time a user can be inactive before the user’s session times out and closes. It only affects user browser sessions.

You can set the values from 5 minutes to 60 minutes.

This function has a default value of 30 minutes. You can set the value up to 60 minutes, however doing so might cause extra load on the system.

**NOTE**

This feature is available as of Platform update 29.

If you previously set a session inactivity timeout in the web.config (`WebClientStatefulSessionTimeoutInSeconds` key) through a support request, then that old value will still be honored. The change in default will only affect those who had not explicitly set a new session inactivity timeout in the web.config.

To change the value, follow these steps:

1. Select **System administration > Setup > System parameters** to open the **System parameters** page.
2. On the **General** tab, in the **Session management** section, enter a value in the **Session inactivity timeout in minutes** field.
3. Select **Save**.

   If you set the value to greater than 30, you will be prompted to confirm your selection. The confirmation prompt says “Increasing the inactivity session timeout can cause extra load on your system, which can lead to a decrease in performance. Are you sure you want to continue?” The higher the value, the higher the load will be, which can affect negatively system performance. Select **Yes** to save the changes, or **No** to revert to the existing value.

### Alerting users before sessions end due to inactivity

To give users awareness of an impending session suspension due to inactivity and to help prevent users from losing any unsaved changes when this occurs, users will be notified before their sessions are set to be terminated due to inactivity and given an opportunity to reconnect. The notice given to the user is dependent on the **Session inactivity timeout** setting.

- If the **Session inactivity timeout** is more than 30 minutes, the user will see a countdown notification starting **5 minutes** before the session is set to close.
- If the **Session inactivity timeout** is between 10 and 30 minutes, the user will see a countdown notification starting **2 minutes** before the session is set to close.
- If the **Session inactivity timeout** is less than 10 minutes, the user will see a countdown notification starting **30 seconds** before the session is set to close.
Finance and Operations provides a set of rich security reports to help you understand the set of security roles running in your environment and the set of users assigned to each role. In addition to the reports noted in this topic, developers can generate a workbook containing all user security privileges for all roles using Visual Studio > Dynamics 365 > Addins > View related objects and licenses for all roles.

Each of the security reports can be found under System administration > Inquiries > Security. A description of each report is provided below.

### User role assignments

The **User role assignments** report generates a view of the current user role assignments in your system. By default, the report includes all users with roles assigned. You can optionally limit the report to a specific set of users by entering a list of users when generating the report. On the **User role assignments** parameters pane, go to Records to include > Filter. From here you can add or remove filters to the list of users the report will be generated for.

<table>
<thead>
<tr>
<th>Organization type</th>
<th>Operating unit types</th>
<th>Organization name</th>
<th>Organization ID</th>
<th>Grant with children</th>
</tr>
</thead>
<tbody>
<tr>
<td>Legal entity</td>
<td>None</td>
<td>ALL</td>
<td>ALL</td>
<td>No</td>
</tr>
</tbody>
</table>

### Role to user assignments

The **Role to user assignment** report provides an aggregation of role assignments. Expanding a role in the
Security role access

The Security role access report provides a view of the effective permissions for each security role. This report provides a flattened list of permissions grouped by type across all sub-roles, duties, and privileges contained in the role.

The data set backing the Security role access report can be very large, causing the report to take some time to run. If there have been no changes to security roles since the last time the report was run, you can skip building the report by setting the Rebuild collection option to No on the report parameters pane. This will render the report from the existing data set. If it is the first time the report has run, or there could be changes to the role definitions, the Rebuild collection option should be set to Yes. You can optionally limit the roles to be included in the report by adding a filter under Records to include.
Expanding a role shows the category of objects the role has access to. Expanding one of the object types will show a detailed list of each object of that type included in the role.

**Security duty assignments**

The **Security duty assignments** report provides a view of all the duties contained within a role. This report can be configured to run on any collection of roles to ensure that segregation of duties is maintained between roles. By default, the report will include all roles. To limit the roles included, leverage the filtering provided in the **Records to include** section.

<table>
<thead>
<tr>
<th>SECURITY ROLE</th>
<th>LICENSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllTestSecRole</td>
<td>Team Members</td>
</tr>
<tr>
<td>Applicant anonymous (external)</td>
<td>None</td>
</tr>
<tr>
<td>Auditor</td>
<td>Operations</td>
</tr>
<tr>
<td>Product designer</td>
<td>Operations</td>
</tr>
<tr>
<td>Product design manager</td>
<td>Operations</td>
</tr>
<tr>
<td>Budget clerk</td>
<td>Activity</td>
</tr>
<tr>
<td>Menu item display</td>
<td></td>
</tr>
<tr>
<td>Menu item output</td>
<td></td>
</tr>
<tr>
<td>Menu item action</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OBJECT</th>
<th>ACCESS</th>
<th>CHILDREN</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCOUNTINGSOURCEEXPORTFILEIMPORT</td>
<td>View</td>
<td></td>
</tr>
<tr>
<td>ASSETBUDGETUPDATE</td>
<td>Full control</td>
<td></td>
</tr>
<tr>
<td>ASSETBUDGETUPDATEANDTRANSFER</td>
<td>Full control</td>
<td></td>
</tr>
<tr>
<td>ASSETBUDGETUPDATEANDTRANSFERDELGER</td>
<td>Full control</td>
<td></td>
</tr>
<tr>
<td>ASSETCONSUMPTIONCREATEPROPOSAL</td>
<td>Full control</td>
<td></td>
</tr>
<tr>
<td>ASSETCONSUMPTIONPROPOSALDELETE</td>
<td>Full control</td>
<td></td>
</tr>
<tr>
<td>ASSETCONSUMPTIONPROPOSALTOJOURNAL</td>
<td>Full control</td>
<td></td>
</tr>
<tr>
<td>ASSETCREATEJOURNAL_DEPRECIATION</td>
<td>Full control</td>
<td></td>
</tr>
<tr>
<td>ASSETCREATEJOURNAL_EXTRAORDINARY</td>
<td>Full control</td>
<td></td>
</tr>
<tr>
<td>ASSETCREATEJOURNALTransferCapital</td>
<td>Full control</td>
<td></td>
</tr>
<tr>
<td>ASSETDELETEBUDGETMODEL</td>
<td>Full control</td>
<td></td>
</tr>
<tr>
<td>BUDGETTRANSACTIONWORKFLOWRECALL</td>
<td>Full control</td>
<td></td>
</tr>
<tr>
<td>BUDGETTRANSACTIONWORKFLOWRESUBMIT</td>
<td>Full control</td>
<td></td>
</tr>
<tr>
<td>BUDGETTRANSACTIONWORKFLOWSUBMIT</td>
<td>Full control</td>
<td></td>
</tr>
<tr>
<td>BUDGETPLANALLOCATE</td>
<td>Create</td>
<td></td>
</tr>
<tr>
<td>BUDGETPLANCOMPLETE</td>
<td>View</td>
<td></td>
</tr>
</tbody>
</table>
Expanding a role in the **Security duty assignments** report will show each duty assigned to the role, along with details of the duty.

**Batch processing of reports**

Any of the above reports can be set to run as a batch job by going to the **Run in the background** section of the report's parameter pane. Set **Batch processing** to **Yes**, then provide a batch task job name, batch group, and whether the job should run as Private or Critical. The report will then be created when the batch task runs.
Before you can access Finance and Operations apps, you must first be added to the Users page (System administration > Users > Users). Users include internal employees of your organization, or external customers and vendors. Users can be imported or added manually. All users must be correctly licensed for compliant use.

For information about how to buy and license for Finance and Operations apps, see Microsoft Dynamics 365 Licensing Guide.

Assign a license to a user

System admins can assign licenses to users in the Microsoft 365 admin center.

Add an external user in Azure AD and assign a license

External users must be represented in your tenant directory (Azure Active Directory (Azure AD)) so that they can be assigned licenses. Those external users should be added to the tenant in Azure AD as guest users and then assigned the appropriate licenses. A requirement for Finance and Operations apps is that the guest user’s company must use Azure AD. For more information, see Add Azure Active Directory B2B collaboration users in the Azure portal.

Import new users from Azure AD

1. Go to System administration > User > Users.
2. On the Action Pane, select Import users.
3. Select the users to be imported. The list includes Azure AD users that are currently not users in this environment.
4. Select Import users.
5. Select Close.

NOTE

The value for the Company field will be set based on the current session company for the admin. After import, you must assign roles and organizations as applicable. For more information, see Assign users to security roles. Conditionally, it might also be required to associate the user with a Person and to update user options such as language.

Manually add a new user

1. Go to System administration > Users > Users.
2. On the Action Pane, select New.
3. In the User ID field, enter a unique identifier for the user.
4. In the User name field, enter the user’s name.
5. In the Provider field:
   - For internal users, use the defaulted value. For example, your Azure AD tenant prefixed with https://sts.windows.net/.
   - For non-Azure AD users, such as Service-2-Service accounts, enter a basic text value. For example, NA.
value will help avoid incorrect authentication calls that might result in errors if a valid identity provider value is used.

- For external or guest users, add their Azure AD tenant name after https://sts.windows.net/.

6. In the Email field, enter the user’s full Email/User Principle Name.
7. In the Company field, select the default startup company for the user.
8. Select Save.

The values for Identity provider and Telemetry ID will be updated based on a Microsoft graph call, when the user record is saved. The Telemetry ID is based on the user’s Object ID/Security Identifier (SID) in Azure AD.

**NOTE**

After you add a user, you must assign roles and organizations as applicable. For more information, see Assign users to security roles. Conditionally, it might also be required to associate the user with a Person and to update User options such as language.

**Change a user ID**

To change a user ID, you must rename the key in the database. When you change a user ID by using this procedure, all related user settings are modified to use the new user ID. For example, the usage information in the SysLastValue table is updated to reference the new user ID.

**NOTE**

The user ID is the primary key of the user information table. Renaming the primary key can take some time for existing users because all references to the key are also updated in the database.

1. Go to System administration > Users > Users.
2. Select a user in the list and select Options > Record info.
3. Select Rename.
4. Enter a new and unique value for the User ID, and then select OK.
5. Select Yes to confirm.

**Additional resources**

For more options to implement B2B users, see Export B2B users to Azure AD.

For information about preconfigured system accounts, see Preconfigured system accounts.
The user session setting represents the amount of time a user can be signed in before the user’s session expires. After the user’s session expires, the user is required to sign in with their credentials.

The **Maximum session length** can be up to 2,160 hours (90 days), with a minimum of 1 hour.

**NOTE**

This feature is available in public preview as of version 10.0.16. To enable this feature, go to the Feature management page and enable the **(Preview) Enable session management for users** feature.

To change the maximum session length, follow these steps:

1. Go to **System administration > Users > User session management**.
2. Select **New**.
3. In the new row, select the drop-down menu in the **User ID** field.
4. In the user list, select a user.
5. In **Maximum session length (hours)**, enter a value.
6. Select **Save**.

To update the user’s maximum session length:

1. Select the user that you want to update by selecting the row.
2. In **Maximum session length (hours)**, enter a value.
3. Select **Save**.

To delete a user’s maximum session length and replace it with another user’s session:

1. Select the user that you want to delete by selecting the row.
2. In the **User ID** field, select the drop-down menu and select another user.
3. In the **Maximum session length (hours)** column, enter an hour.
4. Select **Save**.

To delete the user’s maximum session length:

1. Select the user that you want to delete by selecting the row.
2. Select **Delete**.

To delete the multiple users’ maximum session length:

1. Select the rows that you want to delete.
2. Select **Delete**.
Import users from Azure Active Directory

11/24/2021 • 2 minutes to read • Edit Online

Import select users

This procedure can be used by system administrators to import select users from Azure Active Directory (Azure AD).

1. User will be imported with the current session company as their default company. Change current company if applicable before importing users.
2. Go to System administration > Users > Users.
3. Click Import users.
4. Select the users that should be imported and select Import users.

After import is completed it will be required to assign roles to users.

Import users in bulk

This procedure can be used by system administrators to import a large number of users from Azure Active Directory. Note that it is not possible to select users when using the Batch import option.

Run the import as a batch job

1. User will be imported with the current session company as their default company. Change current company if applicable before importing users.
2. Go to System administration > Users > Users.
3. Click Batch import.
4. Expand the Run in the background section.
5. Select Yes in the Batch processing field.
6. In the Batch group field, enter or select a value. This is an optional step.
7. Select Yes in the Private field. This is an optional step.
8. Select Yes in the Critical job field. This is an optional step.
9. In the Monitoring category field, select an option.
10. Click OK.

After import is completed, it will be required to assign roles to users.

Run in a sandbox environment

1. Select Batch import.
2. Select OK.
You can set up rules to separate tasks that must be performed by different users. This concept is named segregation of duties. For example, you might not want the same person to acknowledge the receipt of goods and to process payment to the vendor. Segregation of duties helps you reduce the risk of fraud, and it also helps you detect errors or irregularities. You can also use segregation of duties to enforce internal control policies. Complete the following procedure to create a rule. You must be a system administrator to complete the procedure.

2. Click New.
3. In the Name field, type a value for the rule.
4. In the First duty field, click the drop-down button to open the lookup.
5. In the list, find and select the desired record. Select the first duty that is controlled by the rule.
6. In the Second duty field, click the drop-down button to open the lookup.
7. In the list, find and select the desired record. Select the second duty that is controlled by the rule.
8. In the Severity field, select an option. Select the severity of the risk that occurs when the same user or role performs both duties.
9. In the Security risk field, type a value. Enter a description of the security risk.
10. In the Security mitigation field, type a value. Enter a description of the actions that you take to mitigate the security risk. For example, you can mitigate the risk by conducting more detailed reviews of the process, by conducting a monthly managerial review, or by sharing resources with other departments.
11. Click Save.

IMPORTANT
Compliance with the rules for segregation of duties is not verified when you create a rule. You can create a rule that creates a conflict for existing roles. Existing user role assignments can also be in conflict with the new rule. You must validate compliance after you create or modify a rule. For more information, see Identify and resolve conflicts in segregation of duties
This topic explains how to identify and resolve conflicts in segregation of duties. You can set up rules to separate duties that must be performed by different users. This concept is named segregation of duties. When the definition of a security role or the role assignments of a user violate the rules, the conflict is logged. All conflicts must be resolved by the administrator. Complete the following procedure to identify and resolve conflicts.

After a rule has been added, verify that all existing roles are compliant.

2. Select Validate duties and roles. If any roles violate the rules, a message is displayed that contains the name of the role, the role, and the names of the conflicting duties. Conflicting roles must be modified using Security configuration and can't include conflicting duties. If no roles violate the selected rule, a message indicates that all roles comply.

**NOTE**
The validation is only performed for the selected rule. It is important to validate compliance for each rule.

When you create or modify a role, the rules for segregation of duties are automatically enforced. You cannot assign conflicting duties to a role.

Next, verify that all existing role assignments are compliant.

1. Go to System administration > Security > Segregation of duties > Verify compliance of user-role assignments.
2. Select OK. A notification displays the results of the validation. Conflicts are logged on the Segregation of duties unresolved conflicts page.

When you assign users to roles, the rules for segregation of duties are automatically enforced. If you try to assign a user to roles that contain conflicting duties, you receive an error message. You must then resolve the conflict by denying or allowing the additional role assignment. The additional role will be assigned after the assignment is allowed.

**NOTE**
Conflicts are currently not verified for users that are assigned roles based on the Active Directory Domain groups.

View and resolve conflicting user role assignments
1. Go to **System administration > Security > Segregation of duties > Segregation of duties unresolved conflicts**.

2. Select a conflict, and then select one of the following actions:

   - **Deny assignment**: This will deny the assignment of the user to the additional security role. If you deny an automatic role assignment, the user is marked as excluded from the role. The excluded user isn't granted the access associated with the role and can't be assigned to the role until the administrator removes the exclusion.

   - **Allow assignment**: This will override the conflict and allow the user to be assigned to the additional security role. If you override a conflict, you must enter a reason in the **Reason for override** field. All overridden role assignments can be viewed on the **Segregation of duties conflicts** page.

   **NOTE**

   If several conflicts are listed for the same user, select the user record and evaluate assigned roles on the **Users** page. To avoid this conflict, validate each rule after it's added or modified.
Assign users to security roles

To use anything other than common capabilities in Finance and Operations apps, users must be assigned to security roles. You can assign users to roles automatically, based on rules and business data, exclude users from automatic role assignment, or add users to roles manually.

Automatically assign users to roles

This procedure explains how system administrators can automatically assign users to roles, based on business data.

1. Go to Navigation pane > Modules > System administration > Security > Assign users to roles.
2. In the tree, select 'Accounting supervisor'. Select the role that you want to configure the rule for. In this example, select Accounting supervisor.
3. Select Add rule to open the dialog menu.
4. In the Select a query list, find and select the desired record. Select the query to use for this rule.
5. In the Membership rule name list, click the link in the selected row.
6. Select Edit query. Edit the query, as needed.
7. Select OK.
8. Select Run automatic role assignment.
10. Review the roles assigned to various users to confirm that the role assignment query was correct. Adjust and re-run if needed.

Exclude users from automatic role assignment

1. Close the page.
2. Go to Navigation pane > Modules > System administration > Security > Assign users to roles.
3. In the tree, select 'Accounting supervisor'. Select a role. For this example, select Accounting supervisor.
4. In the Users assigned to role menu, select Manually assign / exclude users.
5. In the Assign users to or exclude users from role list, mark the selected row. Select a user.
6. On the Action pane, select Exclude from role.
7. Select Exclude from role to exclude the selected users from the role. To remove exclusions, select the users that you want to remove exclusions for, and then click Reset status. When you remove an exclusion by resetting the user’s status, the user’s role is assigned automatically. However, the user is not immediately assigned to the role or excluded from the role when you reset the status. Instead, the user is either assigned to the role or removed from the role the next time that the rules for automatic role assignment are run.

Manually assign users to roles

Users who are manually assigned to security roles must also be manually removed by the administrator. These users are not removed from roles by rules for automatic role assignment.

1. Go to Navigation pane > Modules > System administration > Security > Assign users to roles.
2. In the tree, select a role, and in the Users assigned to role menu, select Manually assign / exclude users.
3. In the **Assign users to or exclude users from role**, users that have not been assigned the role are listed with the **Assignment mode** set to **None**. Select one or more users that should be assigned the role.

4. On the **Action pane**, select **Assign to role**. The **Assignment mode** is updated to **Manual** and the users now have a new role assigned.
Import or export a customized security configuration by using Data management

11/24/2021 • 2 minutes to read • Edit Online

The topic explains how a customized security configuration can be exported and imported across environments by using the Data management framework. This functionality can be used when, for example, a customized security configuration must be moved from a test environment to a production environment.

The following entities hold the customized, role-based security (that is, privileges, duties, and roles) that has been added or modified by using security configuration:

- Security privilege metadata customization entity
- Security duty metadata customization entity
- Security role metadata customization entity

Export customized security configuration

1. Go to System administration > Workspaces > Data management.
2. Select the Export tile.
3. In the Group name field, enter a name for the group.
4. Set the Generate data package option to Yes.
5. Select Add multiple to open the drop-down dialog box.
6. Filter the entities by setting the following fields:
   - In the Entities field, enter Security.
   - In the Entity category field, select Master.

![Export custom security configuration interface](image-url)
7. In the **Target data format** field, select **Excel**.

8. Select the applicable security customization entities.

9. Select **Add selected**.

   In version 10.0.12 and later, ignore any warning messages about data length. Those messages aren't applicable, because the entities that are included use containers in data package mode.

10. Select **Close**.

11. Make sure that the **Sequence** field is set in the order of the entity dependencies. Privileges should be first, then duties, and finally roles.

12. Select **Export**.

13. Select **Close**.

14. Wait for the job to be completed. Select **Refresh** to view the status.

15. Select **Download package**.

16. Save the package.

**Import customized security configuration**

1. Go to **System administration > Workspaces > Data management**.

2. Select the **Import** tile.

3. In the **Group name** field, enter a name for the group.

4. Select **Add file**.

5. Select **Upload and add**.

6. Find the exported package, and then select **Open**.

   In version 10.0.12 and later, ignore any warning messages about data length. Those messages aren't applicable, because the entities that are included use containers in data package mode.

7. Select **Close**.

8. Select **Import**.

9. Select **Close**.

10. Wait for the job to be completed. Select **Refresh** to view the status.

**Related security configuration entities**

- **SystemSecurityUserRoleOrganizationEntity** – Assignment of organizations to security roles.
- **Security segregation of duties rule** – Segregation of duties rules.
- **Security segregation of duties conflict** – Segregation of duties conflicts. This entity has unresolved conflicts but also reviewed conflicts.

**Additional resources**

- [Data import and export jobs overview](#)
- [Move all user and security settings with data entities (blog post)](blog post), by André Arnaud de Calavon
Before you begin

This topic provides information about how to analyze and manage security permission requirements based on a task recording. Before you complete the steps in this topic, you must have a task recording of the business process that you want to analyze. To record a business process, see Task recorder resources.

Manage security for a task recording

2. Open the task recording from its location. Select Open from this PC or Open from Lifecycle Services, and then select Close.
3. This will open the Security menu item details page that lists the security objects required for the process.

**NOTE**
The Action and Output menu items are not included in the list.

4. In the User ID field, select a user. If the user does not have permissions for some menu items, the Missing permissions field will update to Yes.

5. Select Add Reference to see a list of the security objects, including roles, duties, and privileges that grant the missing permission.

6. Select a security object from the list:
   - If Role is selected, select Add role to user. This will open the Assign users to roles page. For more information, see Assign users to security roles page.
   - If Duty is selected, select Add duty to role, select the roles that the duty should be added to, and then select OK.
   - If Privilege is selected, select Add privilege to duties, select the roles that the duty should be added to, and then select OK.
This topic provides an overview of Extensible Data Security (XDS) policies in Finance and Operations apps. XDS allows developers to supplement role-based security by restricting access to table records based on security policies. The query in the policy applies a filter and only records that satisfy the conditions of the filter will be accessible from the restricted tables.

Data security policy components

- **Constrained tables**: The table or tables from which data is filtered or secured. For example, in a policy that secures access to transactions based on customer, the `CustTrans` would be an example of a constrained table.

- **Primary table**: Used to secure the content of the related constrained table. In the example below, the `CustTable` table would be the primary table. The primary table must have an explicit relationship to the constrained tables.

- **Policy query**: Used to secure the constrained tables content using a range condition on the primary table contents. Only records that are included in the range will be accessible. The range can, for example, be based on a specific value for Customer.

- **Context** – Controls the conditions under which a policy is applicable. Two main types of contexts are available:
  - **Role context**: Based on the roles that the user is assigned. There are two sub-options for role context:
    - **RoleName** – Indicates that the security policy is only applied to the application user assigned to the role equal to the value of RoleName.
    - **RoleProperty** – This value is used in combination with the `ContextString` property to specify multiple user roles context. It is applied when the Context String value defined in the `Role Property` field for the policy is the same as the `ContextString` field value for the assigned user roles.
  - **Application context**: Applied if the context string set by the application using the XDS::SetContext API is the same as the value defined in the `Context String` field for the policy.
In the Application Object Tree (AOT), policies and their components are displayed under **Security > Policies**.

### Important considerations

The policy query is added to the WHERE clause, or ON clause, on SELECT, UPDATE, DELETE and INSERT operations involving the specified constrained tables. Unless carefully designed and tested, policy queries can have a significant performance impact. Therefore, make sure to follow simple but important guidelines when developing an extensible data security policy. For more information, see the "Developing efficient extensible data security policies" section in *Developing Extensible Data Security Policies (White paper) [AX 2012]*.

When two or more security policies apply, the intersection (not the union) of the records that are included by each policy are the only records that can be accessed. This means that a record must satisfy all the applicable security policies before access to the record is allowed.

### Additional resources

For information about how to debug policies, create more advanced policies, including chaining of restricted tables, table relations based on expressions and much more please refer to these resources:

- Create a simple security policy
- Developing Extensible Data Security Policies (white paper) [AX 2012]
- Securing Data by Dimension Value by using Extensible Data Security (white paper) [AX 2012]
- Extensible Data Security examples – by Andre Arnaud De Calavon [blog]
- Extensible Data Security (XDS) Framework in D365FO - by Alex Meyer [blog]
This topic explains how to create a simple security policy that secures access to customers and customer groups, based on a range for a customer group.

**Add a new query**

1. In Visual Studio, add a new query, such as XDSQCustGroup10, to your project/solution. The query will be used to restrict data access from the **Constraint** table.

![Add New Item - XDSDemo (SYS) [Application Suite]](image)

2. Right-click **Data Sources**, and select **New Data Source**.
3. In the **Table** field, enter the primary table name **CustGroup**.
4. Right-click **Ranges**, and then select **New Range**.
5. Set the **Enabled** field to **Yes**.
6. In the **Data Source** field, enter the primary table name, in this case, 'CustGroup'.
7. In the **Value field**, enter **10** to restrict access to data where CustGroup has value of 10, by defining the Range for the CustGroup field.
Add a new security policy

1. Add a new security policy, such as XDSCustTableOnCustGroup10.

2. Set **Constrained Table** to **Yes**. This will also secure access to the primary table. In this example this is the **CustGroup** table.

3. Set the **Context Type** field to **RoleName**.

4. Set the **Enabled** field to **Yes**.

5. Set the **Operation** field to **AllOperations**. Other available values for **Operation** include **Select**, **Insert**, **Update**, **Delete**, and **InsertUpdateDelete**.

6. Set **Primary Table** field to **CustGroup**.

7. Set the **Query** field to the name of the query created above, for example ‘XDSQCustGroup10’.

8. Set the **Role Name** field to ‘TradeSalesClerk’. Because **Context Type** is set to RoleName for this policy, it is required to enter the AOT name for a user role.
9. Next, add constrained tables. In this simple example add one table.

   ![Diagram of constrained tables]

   a. Right-click **Constrained tables**, and then select **New > Constrained Table**.

   b. Set **Constrained** to **Yes**.

   c. In the **Name** field, enter the Constrained table, for example ‘CustTable’.

   d. In the **Table Relation** field, enter the relationship to the primary table, in this case ‘CustGroup’.

10. As a final step, it is required that you build and synchronize the solution to activate the policy.
This topic provides an overview of how customers can stay compliant with the user licensing requirements for Finance and Operations apps, such as Microsoft Dynamics 365 Finance, Dynamics 365 Supply Chain Management, and Dynamics 365 Commerce.

The licensing requirements for users are determined by the security roles that are assigned to those users. Security roles are built based on a hierarchy of:

- Sub-roles
- Duties
- Privileges
- Directly referenced securable objects

For more information, see Role-based security.

The licensing requirements for users are determined at the organization or tenant level. This topic is focused on the requirements for a single environment. If you have multiple environments, the requirements must be analyzed across all of them.

A licensing requirement is assigned to every securable object. Examples of licensing requirements include None, Team members, Activity, and Operations. The Operations licensing requirement indicates that a full user license is required. Some privileges are unique to a specific full user license and require a base or attach license for the full user license to be assigned to the user. For more information, see the User license estimator report section later in this topic.

The rest of this topic describes the various tools that you can use to ensure that the actual licensing complies with the expected licensing requirements.

### Roles for selected user FactBox on the Users page

You assign roles to users on the Users page (System administration > Users). You can view license requirements for each role in the Roles for selected user FactBox.

The maximum license requirement determines the actual licensing requirement for a user. If any license...
NOTE
Starting with platform update for version 10.0.15, the required license types are extended to include application specific license types such as Commerce, Finance, and Supply Chain Management. This extension makes it possible to view the actual license requirements when you assign roles to users and when using the Roles for the selected user FactBox. More than one value can be displayed for the required license type.

View permissions page

During security configuration on the Configure security page (System administration > Security > Configure security), you can select any security object, a role, duty, or permissions, and then select View permissions to view all permissions that are currently included and their licensing requirements. The header of the View permissions page shows the required license level.

<table>
<thead>
<tr>
<th>Role</th>
<th>Duty</th>
<th>Privilege</th>
<th>Resource type</th>
<th>Read</th>
<th>Update</th>
<th>Create</th>
<th>Delete</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accounts payable clerk</td>
<td>View vendor invoice entry works...</td>
<td>Vendor invoice entry</td>
<td>Menu item display</td>
<td>Grant</td>
<td>Unset</td>
<td>Unset</td>
<td>Unset</td>
<td>Team Members</td>
</tr>
<tr>
<td>Accounts payable clerk</td>
<td>View vendor invoice entry works...</td>
<td>Open purchase orders list</td>
<td>Menu item display</td>
<td>Grant</td>
<td>Unset</td>
<td>Unset</td>
<td>Unset</td>
<td>Team Members</td>
</tr>
<tr>
<td>Accounts payable clerk</td>
<td>View vendor invoice entry works...</td>
<td>Product receipt list</td>
<td>Menu item display</td>
<td>Grant</td>
<td>Unset</td>
<td>Unset</td>
<td>Unset</td>
<td>Team Members</td>
</tr>
<tr>
<td>Accounts payable clerk</td>
<td>View the discrepancies between...</td>
<td>View approved divergences</td>
<td>Menu item display</td>
<td>Grant</td>
<td>Unset</td>
<td>Unset</td>
<td>Unset</td>
<td>Team Members</td>
</tr>
<tr>
<td>Accounts payable clerk</td>
<td>View the discrepancies between...</td>
<td>Request new approval</td>
<td>Menu item action</td>
<td>Grant</td>
<td>Grant</td>
<td>Grant</td>
<td>Grant</td>
<td>Operations</td>
</tr>
<tr>
<td>Accounts payable clerk</td>
<td>View the discrepancies between...</td>
<td>View the discrepancies between...</td>
<td>Menu item display</td>
<td>Grant</td>
<td>Grant</td>
<td>Grant</td>
<td>Operations</td>
<td></td>
</tr>
<tr>
<td>Accounts payable clerk</td>
<td>View tax document transit relat...</td>
<td>View tax document transit relat...</td>
<td>Table</td>
<td>Grant</td>
<td>Unset</td>
<td>Unset</td>
<td>Unset</td>
<td>None</td>
</tr>
<tr>
<td>Accounts payable clerk</td>
<td>View tax document transit relat...</td>
<td>View tax document transit relat...</td>
<td>Table</td>
<td>Grant</td>
<td>Unset</td>
<td>Unset</td>
<td>Unset</td>
<td>None</td>
</tr>
</tbody>
</table>

NOTE
Starting with platform update for version 10.0.15, the required license types are extended to include application specific license types such as Commerce, Finance, and Supply Chain Management. This extension makes it possible to identify the specific security objects that determine the actual license requirements while also configuring security. More than one value can be displayed for the required license type.

User license counts report

The User license counts report (System administration > Inquiries > License) is used to get a count of required licenses per license type (for example, Team members, Activity, and Operations).
The report also provides details about each user and the licensing requirements for each assigned role. Users are listed under the highest license type. If the license requirement is identified as Operations, you must use the User license estimator report to determine the specific full user licensing requirements.

The User counts history report shows total counts per date, but without any details.

**NOTE**

This report depends on the Named user license count reports processing batch job. To determine when the batch was last run, use the Batch job history page.

**User license estimator report**

If the previously mentioned tools have identified that any users require a license of the Operations type, you can use the User license estimator report (System administration > Inquiries > License reports) to get a count of specific licensing requirements for those users. The report includes only those users.

If no privileges that require a specific full user license have been assigned to a user, that user is shown, but no specific full user licenses are marked. That user will be compliant with any full user license that is assigned. In the example in the following illustration, the Database management administrator role is assigned to a user, and no other roles that require a license of the Operations type are assigned.
Based on the licensing guide, this user is compliant with any full user license.

Admin rights apply across the Finance, Supply Chain Management, and Commerce apps. For example, if you have a Finance license, you have the admin rights for Finance, Supply Chain Management, and Commerce.

This principle might apply to several roles that are provided by Microsoft and customizations, based on the included privileges. No specific license is indicated for these users on the **User license estimator** report.

Alternatively, if privileges that require a specific full user license have been assigned to a user, that license is indicated when you look at the licenses that are assigned to the user. In the example in the following illustration, the user must have a base license for Supply Chain Management.

If privileges that require more than one specific full user license have been assigned to a user, those licenses are indicated when you look at the licenses that are assigned to the user. In this case, the user must have a base license for one of the licenses and an attach license for all other full user licenses that are required. In the example in the following illustration, the user must have either a base license for Finance and an attach license for Supply Chain Management, or a base license for Supply Chain Management and an attach license for Finance.

The totals per specific full user license counts aren’t divided into base licenses and attach licenses.

**NOTE**

The total count per specific full user license and evaluation of custom privileges are included starting in the release of the platform update for version 10.0.13. The report can’t separate the base and attach license requirements. Therefore, it lists only the total full user licenses requirements.

**Additional resources**
For information about how to buy and license Finance and Operations apps, see Microsoft Dynamics 365 Licensing Guide.

For information about how to assign licenses to users in the Microsoft 365 admin center, see Assign licenses to users.

Additional user licenses are required when multiple implementation projects exist for the same tenant. For more information, see Multiple LCS projects and production environments on one Azure AD tenant.
This topic explains how you can view the status of independent software vendor (ISV) licenses for Finance and Operations apps, such as Dynamics 365 Finance, Dynamics 365 Supply Chain Management, and Dynamics 365 Commerce.

License codes and configuration keys are part of the ISV licensing model for Finance and Operations apps.

**NOTE**

Configuration keys that are provided by Microsoft aren't part of the licensing model for Finance and Operations apps. The keys are only used to enable and disable functionality.

When an ISV license key and code are installed, the corresponding configuration key will be available and enabled on the License configuration page.

**View ISV license status**

Each ISV solution that is tied to a license runs only when a valid license code exists in the customer's environment. Therefore, if an ISV ties their solution to a license, but the customer doesn't have a valid license code, the solution doesn't run. To prevent the loss of functionality, it's important to track the expiration dates, if applicable, for license codes. To view the expiration dates, go to System administration > Setup > License configuration, and select the License codes tab.

To avoid downtime, review the expiration dates of the license keys, and if applicable, obtain and import new license keys before moving to a new version. The License codes tab shows the expired license codes. The corresponding configuration key won't appear in the Configuration keys tab.

**Additional resources**

For more information about the ISV licensing feature, see Independent software vendor (ISV) licensing.

For more information about how to import ISV licenses into an on-premises deployment, see Independent software vendor (ISV) licensing (on-premises).

For more information about the configuration keys report, see License codes and configuration keys report.
Process automation allows simple scheduling of processes that will be run by the batch server. The updated calendar view of the scheduled work allows end users to view and take action on scheduled and completed work.

Administration

The central administration page for all process automations is found in the System Administration module under the Setup menu. This page will list all automated processes (series) that are set up in the system. It will also allow you to add new process automations directly from this page. After a series is set up, you can manage each series from this list. You can choose to edit the entire series, delete it, view all occurrences in a list view, or disable the series if you would like to pause the scheduled work for a while.

Any processes that are disabled in feature management won’t show when the feature is disabled. Additionally, the process automation scheduling engine won’t schedule any occurrences or background processes for a disabled feature. Re-enabling the feature will cause any scheduled occurrences or background processes in the past to run immediately. The process automation scheduling engine relies on the system batch job, Process automation polling system job to run. The job shouldn’t be altered or tampered with at any time.

Calendar view

One of the key benefits of process automation is the ability to see the scheduled work in a simple calendar view. This view allows you to see work for a week at a time. You’ll see this view on the right side of the Process automation page. It will be populated with the scheduled work for the selected series.
Occurrence changes

Each occurrence can be modified without impacting other occurrences defined by the series that originated them. Occurrences of scheduled work can be edited from the calendar view by selecting the View/Edit button and selecting Occurrence. This page allows you access to all the settings originally shown in the series setup wizard and provides the ability to make a one-off change for the selected occurrence. An occurrence of scheduled work can also be turned off by selecting the Disable button from the calendar view.

Developer documentation

The process automation framework allows developers to extend the process automation framework. The Process automation framework documentation provides information about how you can create custom processes that you require to be run by the batch server scheduled with the process automation wizard and appear in the calendar view automatically.
This topic provides an overview of batch processing.

Many tasks in Finance and Operations can be run as part of batch jobs. For example, batch jobs can include tasks for printing reports, performing maintenance, or sending electronic documents. By using batch jobs, you can avoid slowing down your computer or the server during typical working hours.

The tasks in a batch job can run either sequentially or at the same time. Additionally, you can create dependencies between tasks. In other words, the sequence of tasks can differ, depending on whether an earlier task succeeds or fails.

You can set up recurrence patterns for batch jobs. For example, you can set up a job to process invoices automatically at the end of every month.

To monitor batch jobs, you can set up alerts. Alerts can be sent when the batch job succeeds, fails, or has finished running.

After a batch job has been processed, you can view the history. The history includes any messages that were encountered while the job was running.

Use batch groups to categorize batch tasks and run them on specific servers. The servers in your environment might have different software installed, or they might be available at different times of the day. Batch groups are used to direct batch tasks to the most appropriate server. Tasks in the same batch job can belong to different batch groups.

For example, server A is set up to print reports, and server B is set up to send electronic documents. You can use batch groups to make sure that reporting tasks are run on server A and electronic documents are processed by server B.

For more information, see Batch processing and batch servers.

**Batch functions**

Administrators and Batch managers can perform common tasks including creating and copying batch jobs, changing a batch job user, and specifying a time period in which a job should not execute. For more information about these tasks, see the following topics:

- Create a batch job
- Batch manager security role
- Active batch periods
- Copy a batch job
- Set up alerts
- Enhanced batch forms
- Clean up the batch job history
- Abort an executing batch job
This topic describes batch processing and batch servers, and how to plan for their use.

The batch framework provides an asynchronous, server-based batch processing environment that can process tasks across multiple instances of Application Object Server (AOS).

You should become familiar with the following aspects of the batch framework:

- **A batch job** is a process that is used to achieve a specific goal. A batch job consists of one or more batch tasks.

- **A batch task** is an activity that is run by a batch job. You can add batch tasks that have multiple types of dependencies to a batch job. You can also configure AOS instances to run multiple threads, each of which runs a task. All batch tasks that are waiting to be run can be run by any available AOS instance that is configured as a batch server. To improve throughput and reduce overall execution time, you can define a batch job as many tasks and then use a batch server to run the tasks against all available AOS instances.

- **A batch group** is an attribute of a batch task. A batch group lets the administrator determine or specify which AOS instance runs the task. When you create a new task, it's put in the default batch group. All batch servers are configured to process the default batch group and the waiting tasks from any job. Additionally, you can create a named batch group, and then set an affinity between that batch group and specific AOS instances. After you create this affinity, only the specified AOS instances will process tasks from the named batch group, and those AOS instances will process tasks from the named batch group only. You can also add the default batch group to the configured servers, if that batch group is required.

### Batch server topology planning

The capacity of a batch server is based on the maximum number of threads that can run concurrently on the AOS instance. Each thread runs one batch task. You can add complex dependencies between or among tasks. You can run these tasks in serial steps or parallel steps, depending on the business logic and requirements. All tasks that don't have any dependencies are considered parallel tasks. AOS instances that are configured as batch servers periodically check for tasks that are waiting to be processed. The batch server assigns each parallel task to a thread and starts to process the thread.

You can run multiple threads across multiple AOS instances. Each AOS instance automatically runs multiple threads, depending on that capacity that is defined in the configuration settings. Therefore, parallel tasks from a job can be run on multiple threads across multiple AOS instances.

A batch server checks for available threads one time per minute. Therefore, you might have to wait for a minute before you see that a waiting task is picked up for processing by an available thread.

### Batch server management planning

All batch servers can be managed from a single location.

One typical use of batch servers is to load balance jobs across multiple servers. You can set the number of threads that the batch server will process.

Because batch servers are also active AOS instances that service requests from the client and other associated components, you must carefully determine when an AOS instance should be available to process batches.
Walkthroughs

The following walkthroughs describe how tasks are processed, and how batch groups can be used to associate batch jobs with batch servers.

**Batch processing of dependent tasks**

For this example, you've created a job that is called JOB 1. As the following diagram shows, the job has seven tasks: TASK 1, TASK 2, TASK 3, TASK 4, TASK 5, TASK 6, and TASK 7.

![Diagram of task dependencies](image)

The tasks have the following dependencies:

- TASK 1 is the first task.
- TASK 2 runs when TASK 1 is completed (regardless of the success or failure of TASK 1).
- TASK 3 runs when TASK 2 is successful.
- TASK 4 runs when TASK 2 is successful.
- TASK 5 runs when TASK 2 fails.
- TASK 6 runs when TASK 3 fails.
- TASK 7 runs when both TASK 3 and TASK 4 are successful.

Two batch servers, Batch1 and Batch2, are configured. Each server has a capacity of one thread.

Imagine that Batch1 checks for waiting tasks, assigns TASK 1 to its thread, and starts to run TASK 1. Although
Batch processing that uses batch groups

NOTE

For this walkthrough, we are using Batch1 and Batch2 to explain the concept. Any batch server that has available threads will start to run a waiting task. You must create a batch group to determine or specify which batch job runs on which server.

Batch processing that uses batch groups

This example shows how batch jobs can be processed on specific batch servers.

You have three batch servers: AOS1, AOS2, and AOS3. By default, all the batch servers process tasks from all batch jobs, depending on the number of available threads.

You create a named batch group, BG1, and configure it to run on AOS2 and AOS3. Therefore, tasks from jobs in BG1 will run only on AOS2 or AOS3, depending on the number available threads. AOS1 won't process tasks from jobs in BG1. Likewise, AOS2 and AOS3 will process tasks from BG1 only.

You can configure AOS2 and AOS3 to process tasks from other batch groups. These batch groups include the default batch group.

Batch excessive tasks configuration (Batch throttling)

Batch throttling can prevent excessive tasks by limiting the average number of executions of a certain batch class per minute. The default upper-bound is 60 tasks per minute. After that, batch framework will suspend the execution of classes for the offending class for another minute, to prevent that specific class from monopolizing the system resources.

NOTE

The batch framework is able to detect instances when there are no non-throttled tasks to be scheduled and executed at any given time. When this occurs, the batch will try to fetch batch tasks from the throttled classes queue to prevent resources from being idle.
Create a batch job

A batch job is a group of tasks that are submitted to an Application Object Server (AOS) instance for automatic processing. Batch jobs are run by using the security credentials of the user who created the job. Use the following procedure to create a batch job. The demo data company used to create this procedure is USMF.

Create the batch job

1. Go to Navigation pane > Modules > System administration > Inquiries > Batch jobs.
2. Click New.
3. In the Job description field, type a value.
4. In the Scheduled start date/time field, enter a date and time.
5. Click Save.

Create a recurrence

1. On the Action Pane, click Batch job.
2. Click Recurrence. Use these options to enter a range and pattern for the recurrence.
3. Click OK.

Add alerts

1. On the Action Pane, click Batch job.
2. Click Alerts. Indicate if you want alert messages sent when the batch job ends, has an error, or is canceled.
   Then specify if you want the alerts to be displayed as pop-up messages.
3. Click OK.

Adjust batch job status

1. Go to System administration > Inquiries > Batch jobs.
2. Select the appropriate batch job.
3. On the Action Pane, click Batch job > Functions > Change status.
4. Select the appropriate status:
   - Withhold: Set the batch job as withhold so it is withheld from the batch job scheduler. Equivalent to stop.
   - Waiting: Set the batch job as waiting so it is waiting to be picked up by the batch job scheduler. Equivalent to go.
5. Click OK.
Enable automatic retries on batch jobs

This topic describes how retries are implemented on batch jobs in Finance and Operations apps, and how you can enable automatic retries on batch jobs when transient failures occur. Currently, if Finance and Operations apps experience a brief loss of connection to Microsoft SQL Server, all batch jobs that are running fail. This behavior disrupts business processes. Because connection loss is inevitable in a cloud service, Microsoft is enabling automated retries when failures of this type occur.

IMPORTANT
This feature is available with version 10.0.18 and later.

Metadata

Because not all batch jobs might be idempotent (for example, when a batch runs credit card transactions), retries can’t be enabled equally across all batch jobs. To help ensure that retries can safely be enabled, Microsoft has added metadata to the batch jobs to indicate whether they can automatically be retried. Between versions 10.0.18 and 10.0.19, more than 90 percent of the Microsoft batch jobs have explicitly implemented the BatchRetryable interface, and the isRetryable value has been set appropriately. For any jobs where the BatchRetryable interface isn’t implemented, the default value of isRetryable is false.

Retries

Retries are enabled only for jobs where the BatchRetryable interface is implemented and isRetryable is set to true. In this new functionality, retries occur on any interruption of the SQL Server connection. Microsoft will continue to add retries on other exceptions.

Frequently asked questions

How do retries work for my custom batch jobs?
There is no change to the custom batch jobs. To take advantage of automated retries, explicitly implement the BatchRetryable interface on the custom batch jobs, and set isRetryable to true. Note that you might have to modify the batch jobs to ensure that they are safe to retry.

How do I implement the retryable interface?
Add the following code to your batch class.

For more information, see Final methods and the Wrappable attribute.
I am extending a Microsoft batch that is retryable, but my batch job isn't retryable. Will the retry be triggered?

Provided that `isRetryable` is set to `false` for the extended job, the job won't be retried.

I mistakenly marked my custom job as retryable. Can I override without taking another code change?

Yes. You can go to `System administration > Setup > Batch class configuration overrides` to unregister your class from being retried when SQL Server connection failures occur, without having to go through a code change. On the `Batch class configuration overrides` page, you must create a new record and add the batch class that you want to unregister. You can use this feature to react quickly to changes that are required. The best practice recommendation is to make sure that the code is updated.

My batch jobs are designed to run multithreaded. How do I implement retries?

If your custom batch process is designed to run in multithreading (that is, if you're creating multiple tasks and adding runtime tasks), you must implement the `BatchRetryable` interface in both the main controller and the task controller.

What is the best practice for the execution time for a batch job?

A batch job that has a shorter execution time is more likely to be successfully completed. Therefore, the need for a retry is avoided.

What is the best practice for the transaction size for a batch job?

A batch job that has a smaller transaction size reduces the amount of work that can be lost because of a transient failure. Therefore, the need for a retry won't drastically increase the total execution time.

What does idempotent mean for a batch job?

In this context, *idempotent* means that a retry won't change or affect the overall result. For example, something should be done only one time and won't be done more than one time. Therefore, something that is done in the original run won't be done again during the retry.

What is the maximum number of retries that `BatchRetryable` supports, and what is the retry interval?

`MaxRetryCount` specifies the number of retries that will be applied to a task, regardless of the type of exception that occurs. If a task fails, the batch platform evaluates the number of times that it has been retried. If the number is less than the value of `MaxRetryCount`, the task is put back into a ready state so that it can be picked up again.

The `BatchRetryable` interface starts after five seconds and stops retrying after the interval time reaches five
minutes. (Interval time increases in the following way: 5, 8, 16, 32, and so on.)

**Can I change the maximum number of retries and the retry interval?**

The **BatchRetryable** interface enables transient SQL connection issues to be handled. It's mainly controlled by the framework. Customers can't update setting for **BatchRetryable**, such as the maximum number of retries and the retry interval.
When you want to create the same jobs for different legal entities, you can use the copy batch job functionality to copy an existing batch job and the batch tasks, including recurrences.

You can set the description, company, schedule start date and time, the recurrence, and the run by account at the same time. When you copy the batch job, any alerts and dependencies from the source job will also be copied.

NOTE
This feature is available as of Platform update 20.

Copy a batch job

Complete the following steps to copy a batch job.

1. Click System administration > Inquiries > Batch jobs.
2. Select the job that you want to copy, and on the Action Pane, click Batch Job > Copy batch job.

4. Enter or add any changes. If you set View tasks to Yes, when you click OK you will go directly to the Batch tasks page for the copied job.
Enable the batch job

Complete the following steps to enable a batch job.

1. On the **Batch job** page, on the Action Pane, click **Batch job > Change status**.
2. Select the **Waiting** status, and then click **OK**.
With the release of Platform update 21, an additional level of control over when batch jobs execute is now available. Previously, it was only possible to schedule a batch job to execute every hour for a specified number of hours or until a given date. Administrators can now provide information for an additional active period, such as in the following scenarios:

- Specifying time ranges during which jobs within a batch group can start execution.
- Selecting to run batch jobs outside of office hours only.
- Setting the recurrence for anytime within the active period. For example, you administrator might select to run the batch jobs every hour, but only between the hours of 6:00 PM and 8:00 AM.

**NOTE**
This feature is available as of Platform update 21.

### Set up active periods for batch jobs

1. Go to **System administration > Setup > Active periods for batch jobs**.
2. Enter the name of the batch job, and specify start and end dates that the batch job is active.
3. Click **Save**.

### Assign active periods to batch jobs

1. Go to **System administration > Inquiries > Batch jobs**.
2. Select the batch job that you want to assign a period to, and click **Edit**.
3. In the **Active period** field, select the active period that you want to assign, and then click **Save**.
Microsoft Dynamics 365 Finance version 10.0.12 includes a Daylight Saving Time support for batch job active periods feature that can be turned on in Feature management. This feature introduces daylight saving time (DST) support for the active periods for batch jobs and lets users associate their active periods with different time zones.

**NOTE**

This feature is a one-way feature. In other words, it can't be turned off after it's turned on.

When this feature is turned on, the following changes occur:

- On the Active periods for batch jobs page, a Timezone field is added for each active period. This field specifies the time zone that the active period uses. By default, every active period initially uses the Coordinated Universal Time (UTC) time zone.

- The start and end times of existing active periods are adjusted according to the UTC time zone. Although the active periods will continue to start and end at the same times that they previously started and ended, the times that are shown might change if the user's preferred time zone isn't UTC.

- Active periods will follow the DST adjustments of the time zones that they are associated with.
Batch manager security role

Before Platform update 20, users needed to be assigned to the system admin or IT admin security role to manage batch jobs. With the release of Platform update 20, there is a more targeted role, Batch manager. With this security role, a user now has permissions to copy batch jobs, change who will execute jobs, and specify the time ranges during which jobs can execute. The Batch maintain security privilege is part of the Batch manager security role and it allows a user to create an ad hoc batch job and grant privileges to other users.

NOTE
This feature is available as of Platform update 20.

Assign the Batch manager role to a user

Complete the following steps to assign the Batch manager security role to a specific user.

1. Select System administration > Security > Assign users to roles.

![Assign users to roles screenshot](image1)

2. Select Batch Job Manager, and on the left pane, select Manually assign/exclude user.

![Manually assign/exclude user screenshot](image2)

3. Select a user from the list, and then select Assign to role.
4. Close the page.

Run by user

The run by user functionality allows Batch managers to specify a user to run the batch job. This functionality is useful when you want to change the user who is currently assigned to run the job or if you want to quickly set a
user while copying the batch jobs from one company to another. You can also use this functionality to copy batch jobs.

<table>
<thead>
<tr>
<th>Progress</th>
<th>Created by</th>
<th>Run by</th>
<th>Company acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>CHARLIE</td>
<td>BENJAMIN</td>
<td>usmf</td>
</tr>
<tr>
<td>0.00</td>
<td>SARA</td>
<td></td>
<td>usmf</td>
</tr>
<tr>
<td>0.00</td>
<td>Admin</td>
<td></td>
<td>usmf</td>
</tr>
</tbody>
</table>
Alerts form a notification system for critical events in Finance and Operations. You can use alerts to stay informed about events that you want to track during the workday. You can set up a set of alert rules so that you're alerted when a batch job ends, ends in error, or is canceled. You can select whether the alerts are emailed to you or appear as notifications in the Action center. Alerts can be set up per batch job and per user.

**Set up alerts for batch enhanced forms**

Follow these steps to set up alerts for batch enhanced forms.

1. Go to **System administration** > **Inquiries** > **Batch jobs**.

2. Select a batch job in the list, and then, on the Action Pane, select **Alerts**.

3. In the **Batch job alerts** dialog box, configure the alerts, and then select **OK**.

4. Check the Action center for alert notifications.
Set up alerts for batch legacy forms

Follow these steps to set up alerts for batch legacy forms.

1. Go to System administration > Inquiries > Batch jobs.

2. Select a batch job in the list, and then, on the Action Pane, on the Batch job tab, select Alerts.

3. In the Batch job alerts dialog box, configure the alerts, and then select OK.

NOTE
To receive email notifications, in the Batch job alerts dialog box, set the Email option to Yes.
You can open an enhanced detail transaction form by selecting the job ID for a batch job. The enhanced form provides a header and lines that summarize the batch tasks and constraints that are related to the selected batch job.

Switch to the enhanced form

Follow these steps to switch to the enhanced form.

1. Go to System administration > Inquiries > Batch jobs.

You're notified about the enhanced form. The notification shows the location of the Switch to enhanced form button on the Action Pane.

2. Select Switch to enhanced form.

To switch back to the unenhanced form, select Switch to legacy form on the Action Pane of the enhanced form.
When you run a batch job, a history is recorded. This history can be used to monitor the correct execution of jobs. However, when several batch jobs have been created, especially batch jobs that have a high recurrence, lots of batch job history entries are generated. Too many entries in the history table can negatively affect the performance of future jobs.

Two pages that have been added to the System administration module make it easy to clean up the batch job history:

- Batch job history clean-up
- Custom batch job history clean-up

We recommend that you regularly clean up the batch job history, and that you do this cleanup outside of business hours.
Follow these steps to quickly clean up all history entries that are older than a specified number of days.

1. On the Periodic tasks in System administration module, select Batch job history clean-up.
2. In the History limit (days) field, specify the number of days to keep a history of batch jobs.
3. Select OK.

**Batch job history clean-up.**

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>History limit (days)</td>
</tr>
</tbody>
</table>

**Run in the background**

Batch job history clean-up (custom)

The custom batch job lets you to apply additional filtering, based on criteria such as status, job description, company, or user. You can also add other filter criteria by selecting the **Filter** button.

1. On the Periodic tasks in System administration module, select Batch job history clean-up (custom).
2. In the History limit (days) field, specify the number of days to keep a history of batch jobs.
3. On the Records to include FastTab, specify any filter criteria that you require, and then select **OK**.
4. Select **OK**.
# Batch job history clean-up (custom)

## Parameters

**History limit (days)**

| Value | 30 |

## Records to include

### Filter

**BATCH JOBS HISTORY**

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td></td>
</tr>
<tr>
<td>Job description</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>End date/time</td>
<td></td>
</tr>
<tr>
<td>Company</td>
<td></td>
</tr>
<tr>
<td>Created by</td>
<td></td>
</tr>
</tbody>
</table>

## Run in the background

<table>
<thead>
<tr>
<th>Value</th>
<th></th>
</tr>
</thead>
</table>
NOTE
This feature is available as of Platform update 27.

Sometimes canceling a batch job can take a long time if already executing tasks will take a long time to finish. This option provides a system administrator or batch job manager with the ability to cancel already executing tasks for jobs that are in the process of being canceled. This provides a much faster mechanism to cancel a long running job that is impacting system usage elsewhere.

NOTE
It is important to note that this feature should be used with caution. When you cancel a running process, it is an inherently unsafe action that can lead to data corruption, resulting in either orphaned or incomplete data. This action should only be used to mitigate other issues caused by the running tasks.

Complete the following steps to immediately cancel the running task.

1. Go to System administration > Inquiries > Batch jobs.
2. Select a batch job that has a Status of Canceling.
3. On the Batch tasks tab, select Abort on the task, and then select OK.

Enhanced cancellation feature

Starting in version 10.0.16, an enhancement to the batch cancellation functionality has been introduced. Upon confirmation, this will restart the batch server currently running the batch tasks that you are attempting to cancel. This makes the functionality more resilient to limitations, and ensures that tasks of the job you are trying to cancel are truly preempted.

To use the new functionality, refer to the following steps:

1. Make sure you are running version 10.0.16 or later, or have the necessary quality package installed.
2. Enable the Enhanced batch abort feature in the Feature management workspace.
3. Follow the same instructions to cancel an executing batch job.

You will be prompted that the batch server, which is running the canceling tasks, will be restarted. This can potentially disrupt a list of other batch jobs. You must proceed in order to end the canceling tasks.
If you do not want to cancel other running batch jobs on the server and would prefer the old behavior of canceling a single task and not all the jobs running, you can turn off the Enhanced batch abort feature in the Feature management workspace and try to cancel the executing batch job again.
This topic provides information about the Batch OData application programming interface (API) and explains how you can use Open Data Protocol (OData) to reschedule a job.

In the existing Finance and Operations batch processing functionality, if some types of job errors can be retried (either with or without any changes, based on the interpretation of the error), they must be manually rerun from the batch. For jobs that are scheduled to be run during off-peak times to avoid active business hours for customers, monitoring failures and re-triggering the jobs requires either 24/7 support or a wait time until users resume work during normal business hours.

Current available automation (business events integration)

Business event capabilities enable customers to configure notifications about changes in state (started, failed, finished, or canceled) for batch jobs. Integration with Microsoft Power Automate lets customers capture information about affected jobs without having to sign in to the system. However, manual intervention is required if any action must be taken based on the business events.

For information about how to configure batch events, see Batch business events.

End-to-end automation

In version 10.0.22, the batch functionality now exposes an OData API that can be used to requeue batch jobs. Customers can use the OData endpoint to requeue batch jobs that are in a terminal state. This feature can be integrated with any automation by using Power Automate, custom APIs, and so on.

Automate requeuing of failed batch jobs by using OData API

The Batch OData endpoint lets users consume and automate the end-to-end process to reschedule a batch job by using Power Automate or custom API. It supports updates of the batch job status from a started, failed, finished, or canceled state to a waiting state, based on business requirements.

- **Service endpoint:**
  https://<org url>/data/BatchJobs/Microsoft.Dynamics.DataEntities.SetBatchJobToWaiting

- **Method type:** POST

- **Header:**
  - **Authorization:** Bearer <Bearer token for authentication>
Content-Type: application/json

Body:

```
{
   "batchJobId":<BatchJobId>
}
```

Sample response:

```
{
   "ResponseStatusCode":200,
   "IsSuccess":true,
   "Batch JobId":<BatchJobId>,
   "ExceptionDetails":",
   "reponseMessage":"Status of supplied BatchJobId: *********** is Successfully updated to waiting state"
}
```

Here is an explanation of the elements of the response output:

- **ResponseStatusCode** – A standard HTTP response code, based on the execution of the action.
- **IsSuccess** – A Boolean value that indicates overall success or failure.
- **BatchJobId** – The ID of the input batch job.
- **ExceptionDetails** – Details about any exception that occurred during execution.
- **ReponseMessage** – The success message.
This topic describes how you can use Optimization advisor to help ensure optimal configuration of Finance and Operations.

**Overview**

Incorrect configuration and setup of a module can adversely affect the availability of application features, system performance, and the smooth operation of business processes. The quality of business data (for example, the correctness, completeness, and cleanliness of the data) also affects system performance, and an organization's decision-making capabilities, productivity, and so on.

The **Optimization advisor** workspace is a tool that lets power users, business analysts, functional consultants, and IT support functions identify issues in module configuration and business data. Optimization advisor suggests best practices for module configuration and identifies business data that is obsolete or incorrect.

Optimization advisor periodically runs a set of best practice rules. A default set of rules is available, however users can also create rules that are specific to their customizations, solutions from independent software vendors (ISVs), and business data. For more information about how to create rules, see [Create rules for Optimization advisor](#).

When a violation of a rule is detected, an optimization opportunity is generated and appears in the **Optimization advisor** workspace. A user can take appropriate corrective action directly from the **Optimization advisor** workspace.

Opportunities can be company-specific or cross-company, depending on the type of setup and data that is being validated. Cross-company opportunities can be viewed from all companies. To view the opportunities for a specific company, you must first select the company.

Standard security policies apply to optimization opportunities. For example, the optimization opportunities that are related to configuration of the **Warehouse management** module are visible only to users who have access to Warehouse management and can change its setup.

When you take action on some optimization opportunities, the system calculates the impact of the opportunity in terms of the reduction in the runtime of business processes. Unfortunately, this feature isn't available for all optimization opportunities.

To learn more about Optimization advisor, watch the short [Optimization advisor in Dynamics 365 for Finance and Operations video](#).

**Optimization rules**

To view the complete list of Optimization advisor rules and to see how often the rules are evaluated, go to **System administration > Periodic tasks > Maintain diagnostics validation rule**. Only rules that have a status of **Active** are evaluated. The evaluation frequency can be set to **Daily**, **Weekly**, **Monthly**, or **Unscheduled**.

To trigger the evaluation of unscheduled rules, or to reevaluate periodic rules outside their predefined schedule, go to **System administration > Periodic tasks > Schedule diagnostics validation rule**. Then, in the **Diagnostic rule validation** dialog box, select an evaluation frequency. All rules that have the specified frequency will be reevaluated.
The current set of optimization rules can be divided into the following categories.

**Module configuration and setup**

The setup of Warehouse management is a complicated process. To make the process easier, some rules have been introduced to help validate the correctness of the setup. For example, one rule validates the setup of warehouse location directives for fixed product variant locations for sales orders and transfer orders.

Additionally, some rules check whether features that have been enabled are actually used. For example, one rule determines whether you’re using the Master planning module. If the rule determines that you aren’t using the module, an optimization opportunity is generated to suggest that you turn off the planning processes.

**System configuration**

If specific functionality that is controlled by a configuration key isn’t used, an optimization opportunity is generated to suggest that you disable the configuration key. Examples of configuration keys include Catch weight, Budget planning, Project, and Approved vendor list.

**Business data consistency and cleanup**

If master data isn’t correct (for example, if you have unit of measure conversions for units that haven’t been defined, or if you have unit of measure conversions that have a division by 0 [zero]), an optimization opportunity is generated to suggest that you correct the data.

If you have too many batch job history entries, obsolete items, closed on-hand entries for warehouse enabled items, and so on, or if those entries and items are too old, optimization opportunities are generated to suggest that you clean up the data. By keeping your data clean, you can help improve overall system performance.

**Best practices**

If you aren’t running some business processes according to best practices (for example, if you run inventory pre-closing before the inventory is closed, or if you use the scheduled batch for subledger journal batch transfer), optimization opportunities inform you about the best practice and ask that you follow it.

**Optimization opportunities**

To view the optimization opportunities that are generated during the evaluation of optimization rules, open the Optimization advisor workspace.

In this workspace, you can view more information about an opportunity by selecting More information. If you want the system to take action and correct the setup, clean the data, and so on, so that you don’t have to open the corresponding pages yourself, select Take action.

There is no workflow for optimization opportunities. After you select Take action or use a navigation path that is provided in the More information dialog box, the optimization opportunity disappears from the list. If the corrective action doesn’t completely resolve an issue, the opportunity will be generated again the next time that the rule is evaluated.

If an opportunity doesn’t apply to your role, you can select Hide from my list. Even if the rule behind this opportunity is triggered again later, you won’t see the opportunity in your list.

To deactivate the evaluation of specific rules, select the opportunity that was generated by the rule, and then select Deactivate analysis.

**Additional resources**

Create rules for Optimization advisor

Optimization advisor in Dynamics 365 for Finance and Operations (Video)
This topic explains how to create new rules for Optimization advisor. For example, you can create a new rule that identifies which Request for Quotations (RFQ) cases have an empty title. Using titles on cases makes them easily identifiable and searchable. While quite simple, this example shows what can be achieved with optimization rules.

A rule is a check on application data. If the condition that the rule evaluates is met, opportunities to optimize processes or improve data are created. The opportunities can be acted upon and, optionally, the impact of the actions can be measured.

To create a new rule for the Optimization advisor, add a new class that extends the SelfHealingRule abstract class, implements the IDiagnosticsRule interface, and is decorated by the DiagnosticRule attribute. The class must also have a method decorated with the DiagnosticsRuleSubscription attribute. By convention, that is done on the opportunityTitle method, which will be discussed later. This new class can be added to a custom model with a dependency on the SelfHealingRules model. In the following example, the rule being implemented is called RFQTitleSelfHealingRule.

```java
[DiagnosticsRule]
public final class RFQTitleSelfHealingRule extends SelfHealingRule implements IDiagnosticsRule
{
    
}
```

The SelfHealingRule abstract class has abstract methods that must be implemented in inheriting classes. The core is the evaluate method, which returns a list of the opportunities identified by the rule. Opportunities can be per legal entity or can apply to the whole system.
protected List evaluate() {
    List results = new List(Types::Record);
    DataArea dataArea;
    while select id from dataArea
        where !dataArea.isVirtual
    {
        changecompany(dataArea.id)
        {
            container result = this.findRFQCasesWithEmptyTitle();
            if (conLen(result) > 0)
            {
                SelfHealingOpportunity opportunity = this.getOpportunityForCompany(dataArea.Id);
                opportunity.EvaluationState = SelfHealingEvaluationState::Evaluated;
                opportunity.Data = result;
                opportunity.OpportunityDate = DateTimeUtil::utcNow();
                results.addEnd(opportunity);
            }
        }
    }
    return results;
}

The method shown above loops over companies and selects RFQ cases with empty titles in the findRFQCasesWithEmptyTitle method. If at least one such case is found, then a company-specific opportunity is created with the getOpportunityForCompany method. Notice that the field Data in the SelfHealingOpportunity table is of type Container, and can therefore contain any data relevant to the logic specific to this rule. Setting OpportunityDate with the current timestamp registers the time of the latest evaluation of the opportunity.

Opportunities can also be cross-company. In this case, the loop over companies is not necessary and the opportunity must be created with the getOpportunityAcrossCompanies method.

The following code shows the findRFQCasesWithEmptyTitle method, which returns the IDs of the RFQ cases that have empty titles.

private container findRFQCasesWithEmptyTitle()
{
    container result;
    PurchRFQCaseTable rfqCase;
    while select RFQCaseId from rfqCase
        where rfqCase.Name == ''
    {
        result += rfqCase.RFQCaseId;
    }
    return result;
}

Two more methods that must be implemented are opportunityTitle and opportunityDetails. The former returns a short title for the opportunity, the latter returns a detailed description of the opportunity, which can also include data.

The title returned by opportunityTitle appears under the Optimization opportunity column in the Optimization advisor workspace. It also appears as the header of the side pane showing more information.
about the opportunity. By convention, this method is decorated with the `DiagnosticsRuleSubscription` attribute, which takes the following arguments:

- **Diagnostics area** – An enum of type `DiagnosticsArea` that describes what area of the application the rule belongs to, such as `DiagnosticsArea::SCM`.

- **Rule name** – A string with the rule name. This will appear under the Rule name column in the Diagnostics validation rule form (DiagnosticsValidationRuleMaintain).

- **Run frequency** – An enum of type `DiagnosticRunFrequency` that describes how often the rule should be run, such as `DiagnosticRunFrequency::Daily`.

- **Rule description** – A string with a more detailed description of the rule. This will appear under the Rule description column in the Diagnostics validation rule form (DiagnosticsValidationRuleMaintain).

The `DiagnosticRuleSubscription` attribute is required for the rule to work. Typically, it is used on `opportunityTitle`, but it can decorate any method of the class.

The following is an example implementation. Raw strings are used for simplicity, but a correct implementation requires labels.

```java
[DiagnosticsRuleSubscription(DiagnosticsArea::SCM,
    'Assign titles to Request for Quotation cases',
    DiagnosticRunFrequency::Daily,
    'This rule detects Requests for Quotation with empty titles.')]
public str opportunityTitle()
{
    return 'Assign titles to Request for Quotation cases';
}
```

The description returned by `opportunityDetails` appears on the side pane showing more information about the opportunity. This takes the `SelfHealingOpportunity` argument, which is `Data` field that can be used to provide more details about the opportunity. In the example, the method returns the IDs of the RFQ cases with an empty title.

```java
public str opportunityDetails(SelfHealingOpportunity _opportunity)
{
    str details = '';
    container opportunityData = _opportunity.Data;
    int affectedRFQCasesCount = conLen(opportunityData);

    if (affectedRFQCasesCount != 0)
    {
        details = 'The following Request for Quotation cases have an empty title:
';
        for (int i = 1; i <= affectedRFQCasesCount ; i++)
        {
            PurchRFQCaseId rfqCaseId = conPeek(opportunityData, i);
            details += rfqCaseId + '
';
        }
    }

    return details;
}
```

The two remaining abstract methods to implement are `provideHealingAction` and `securityMenuItem`.

`provideHealingAction` returns true if a healing action is provided, otherwise, it returns false. If true is returned,
the method `performAction` must be implemented, or an error will be thrown. The `performAction` method takes a `SelfHealingOpportunity` argument, in which the data can be used for the action. In the example, the action opens the `PurchRFQCaseTableListPage`, for manual correction.

```java
public boolean providesHealingAction()
{
    return true;
}

protected void performAction(SelfHealingOpportunity _opportunity)
{
    new MenuFunction(menuItemDisplayStr(PurchRFQCaseTableListPage), MenuItemType::Display).run();
}
```

Depending on the specifics of the rule, it might be possible to take an automatic action using the opportunity data. In this example, the system could generate titles for RFQ cases automatically.

`securityMenuItem` returns the name of an action menu item such that the rule is only visible to users who can access the action menu item. Security might require that specific rules and opportunities are accessible only to authorized users. In the example, only users with access to `PurchRFQCaseTitleAction` can view the opportunity. Notice that this action menu item was created for this example, and was added as an entry point for the `PurchRFQCaseTableMaintain` security privilege.

```
public MenuName securityMenuItem()
{
    return menuItemActionStr(PurchRFQCaseTitleAction);
}
```

The method `performAction` must be implemented, or an error will be thrown. The `performAction` method takes a `SelfHealingOpportunity` argument, in which the data can be used for the action. In the example, the action opens the `PurchRFQCaseTableListPage`, for manual correction.

```
public boolean providesHealingAction()
{
    return true;
}

protected void performAction(SelfHealingOpportunity _opportunity)
{
    new MenuFunction(menuItemDisplayStr(PurchRFQCaseTableListPage), MenuItemType::Display).run();
}
```

Depending on the specifics of the rule, it might be possible to take an automatic action using the opportunity data. In this example, the system could generate titles for RFQ cases automatically.

`securityMenuItem` returns the name of an action menu item such that the rule is only visible to users who can access the action menu item. Security might require that specific rules and opportunities are accessible only to authorized users. In the example, only users with access to `PurchRFQCaseTitleAction` can view the opportunity. Notice that this action menu item was created for this example, and was added as an entry point for the `PurchRFQCaseTableMaintain` security privilege.

```
public MenuName securityMenuItem()
{
    return menuItemActionStr(PurchRFQCaseTitleAction);
}
```

After the rule has compiled, execute the following job to have it display in the user interface (UI).

```
class ScanNewRulesJob
{
    public static void main(Args _args)
    {
        SysExtensionCache::clearAllScopes();
        var controller = new DiagnosticsRuleController();
        controller.runOperation();
    }
}
```

The rule will display in the Diagnostics validation rule form, available from System administration > Periodic tasks > Maintain diagnostics validation rule. To have it evaluated, go to System administration > Periodic tasks > Schedule diagnostics validation rule, select the frequency of the rule, such as Daily. Click OK. Go to System administration > Optimization advisor to view the new opportunity.

The following example is a code snippet with the skeleton of a rule including all the required methods and attributes. It helps you get started with writing new rules. The labels and action menu items that are used in the example are only used for demonstration purpose.
public final class SkeletonSelfHealingRule extends SelfHealingRule implements IDiagnosticsRule
{

    public str opportunityTitle()
    {
        // Return a label with the title of the opportunity
        return "$SkeletonRuleLabels:SkeletonOpportunityTitle";
    }

    public str opportunityDetails(SelfHealingOpportunity _opportunity)
    {
        str details = "";
        // Use _opportunity.data to provide details on the opportunity
        return details;
    }

    protected List evaluate()
    {
        List results = new List(Types::Record);
        // Write here the core logic of the rule
        return results;
    }

    public boolean providesHealingAction()
    {
        return true;
    }

    protected void performAction(SelfHealingOpportunity _opportunity)
    {
        // Place here the code that performs the healing action
        // To open a form, use the following:
        // new MenuFunction(menuItemDisplayStr(SkeletonRuleDisplayMenuItem), MenuItemType::Display).run();
    }

    public MenuName securityMenuItem()
    {
        return menuItemActionStr(SkeletonRuleActionMenuItem);
    }
}

For more information, watch the short YouTube video: Optimization advisor in Dynamics 365 for Finance and Operations
Lifecycle Services (LCS) has a feature called **Report production outage**. This feature is available to all customers who have purchased one or more Dynamics 365 Finance and Operations apps and have implementation projects with a production environment deployed in LCS. This feature provides a quick and effective channel to escalate issues to Microsoft Support in the event that the services in a production environment are degraded or become unavailable.

Following mutually inclusive conditions, a production outage can be defined as one or more system-wide issues on a live production environment that impact multiple users and prevent your business from performing daily operations.

**Reporting flow**

The following list shows the order in which an issue should be handled:

1. In a live production environment, a customer experiences an outage or other situation with prevents business from continuing.
2. The customer reports a production outage issue by using the LCS Support portal.
3. The customer selects a production outage issue and provides additional information.
4. A Microsoft support engineer acknowledges the production outage ticket within 30 minutes of submission and begins to immediately collaborate with stakeholders to investigate and resolve the issue.
5. A support engineer contacts the customer to provide a status update.

**Access and availability**

All users who have been added to a customer’s implementation project have access to this feature. This includes project owners, organization admins, team members, and environment managers.

This feature is available for:

- Dynamics 365 Finance
- Dynamics 365 Supply Chain Management
- Environments that are managed by Microsoft
- A production environment in the LCS project
- All support plans

**Report a production outage**

To report a production outage, follow these steps:

1. Log in to your LCS project.
2. From the hamburger menu, click **Support**.
3. On the **Submitted To Microsoft** tab, click **Report production outage**.

4. Confirm the production outage, select the outage scenario from the drop-down list, and then click **Continue**.
5. Add a title and details about the outage, and then click **Next**.
6. Provide contact information, and then click **Next**.
7. Click **Done**.

If you're unable to report a production outage in LCS, **phone support** is available.

**NOTE**

If you don't see your situation listed in the outage scenarios, enter a support incident through LCS. During the initial investigation by a Microsoft support engineer, if it is found that the situation does not meet the current list of production outage scenarios, the support incident will be transferred to the correct support team and service-level agreement (SLA) based on your current support plan.
Database logging provides a way to track specific types of changes to the tables and fields in Finance and Operation apps. Changes that can be tracked include insert, update, delete, and rename key operations. When you configure logging for a table or field, a record of every change to that table or field is stored in the database log table, sysdatabaselog, in the environment database.

Database logging can be used for these purposes:

- Create an auditable record of changes to specific tables that contain sensitive information.
- Monitor the use of electronic signatures. By default, all transactions that have been signed by using electronic signatures are logged.

Database logging is intended to track individual transactions. It isn't intended to track automated transactions that are run in batch jobs.

### Security for database logging

Database logs can contain sensitive data. By default, any user who has database access can query the database log table (sysdatabaselog) by using X++ or alerts, or by querying the database directly. To help protect data, you should restrict permissions on the sysdatabaselog table for on-premises deployments.

### Database logging and performance

Although database logging can be valuable from a business perspective, it can be expensive with regard to resource use and management. Here are some of the performance implications of database logging:

- The database log table can grow quickly and can increase the size of the database. The amount of growth depends on the amount of logged data that you decide to retain.
- When logging is turned on for a transaction type, each instance of that transaction type causes multiple records to be written to the Microsoft SQL Server transaction log file. Specifically, one record is written for the initial transaction, and one record logs the transaction in the database log table. Therefore, the transaction log file will grow more quickly and might require additional maintenance.
- Database logging can adversely affect long-running automated processes, such as inventory close, calculations for bills of materials (BOMs), master planning, and long-running data imports.
- When logging is turned on for a table, all set-based database operations are downgraded to row-based operations. For example, if you're logging inserts for a table, each insert is done as a row-based insert.

Here are some practices that Microsoft recommends:

- Create a plan for how long you will retain logged data, and how you will archive or delete data.
- Limit log entries, and help improve performance by selecting specific fields to log instead of whole tables.

**NOTE**

Only updates can be logged for individual fields.

- After you configure database logging, consider increasing the frequency of backups of the SQL Server transaction log.
Set up database logging

You can use the Logging database changes wizard to set up database logging. This wizard provides a flexible way to set up logging for tables or fields.

1. Go to System administration > Setup > Database log > Database log setup.
2. Select New to open the Logging database changes wizard.
3. Complete the wizard.

Clean up database logs

You can delete database logs as required. You can delete logs for specific tables, delete specific types of database logs, or delete logs based on the date and time when they were created.

1. Go to System administration > Inquiries > Database > Database log.
2. Select Clean up log.
3. Select the method that should be used to select the logs that are deleted. Enter the table ID that the logs refer to, the type of log, or the creation date and time.
4. Use the Database log cleanup tab to specify when the log cleanup task should be run.

Consistency check for database log triggers

In Platform update 34, functionality for a consistency check was added. The consistency check is run as part of the Database log wizard. It’s run after you select Finish or after you select Consistency check on the Database log setup page.

The consistency check will re-create any missing database log triggers. It will also drop any “orphaned” database log triggers that no corresponding configuration is found for. In this way, the consistency check quickly detects and fixes any inconsistencies between the current configuration and the database triggers that are used to implement the logging functionality.

1. Go to System administration > Inquiries > Database > Database log.
2. On the Database log page, select Consistency check.
With the release of Platform update 32, we have introduced the ability to build OData metadata cache when the Application Object Server (AOS) starts, instead of when the first OData request is made. This significantly decreases the response time for the first OData call after an AOS process restart.

This option is useful if your business process can't wait for the OData metadata cache to be built each time that the AOS process restarts. Follow these steps to turn on this feature.

1. Go to System administration > Setup > System parameters.
2. On the General Tab, select Build metadata cache when AOS starts, and then select Save.

**NOTE**

When you enable this functionality, the AOS should already be running and should have served one OData request. This means that the cache is already built. This new functionality will take effect during the next AOS restart.
The functionality for storing advanced certificates lets you define the type of certificate storage that is used in Finance and Operations apps.

The functionality provides two options for storing certificates: local storage and Microsoft Azure Key Vault storage. You can define the option that is used by setting the new *Use advanced certificate store* option on the General tab of the System parameters page (System administration > Setup > System parameters).

- **Local storage** – This storage option can be used with on-premises deployments and any kind of on-premises development environment. To use it, set the *Use advanced certificate store* option to **No**. This storage option is recommended for development environments that are used for development and validation purposes, where it's necessary to validate the certificate and work with it.

- **Azure Key Vault storage** – This storage option is required for cloud deployments, but it can also be used with on-premises deployed environments and any kind of on-premises development environment. To use it, set the *Use advanced certificate store* option to **Yes**. This storage option is the only option for a production environment in the Azure cloud.

Some setup is required before you can work with certificates that are stored in Key Vault. For information about the required settings, see the following Microsoft Knowledge Base (KB) article: 4040294 - Maintaining Azure Key Vault storage. After you set up the Key Vault storage, you should link to the certificates in Finance and Operations apps.

After the certificate is installed in Key Vault, it must be set up in the application.

1. Go to **System administration > Setup > Key Vault parameters**.
2. Select **New** to create a new instance.
3. Enter a name and description, and then, on the **General** FastTab, set the fields that are required for the integration with Key Vault storage:
   
   - **Key Vault URL** – Enter the default Key Vault URL if it isn't already defined by the secret reference.
   
   - **Key Vault client** – Enter the interactive client ID of the Azure Active Directory (Azure AD) application that is associated with the Key Vault storage for authentication.
   
   - **Key Vault secret key** – Enter the secret key that is associated with the Azure AD application that is used
for authentication with the Key Vault storage.

NOTE
If several Key Vault storages are used, you should set up a separate instance for each instance on the Key Vault parameters page.

4. On the Certificates FastTab, select Add to add your certificates. For each certificate, set the following fields:

- **Name**
- **Description**
- **Key Vault certificate secret** – Enter a secret reference to the certificate.

The format of a Key Vault certificate secret must resemble the following example:

```
vault://<KeyVaultName*>/<SecretName>/<SecretVersion*>
```

Attributes that are marked with an asterisk (*) are optional. However, the `<SecretName>` attribute is required.

In most cases, you can define a Key Vault secret key in the following format:

```
vault://<SecretName>
```

If the secret version isn’t defined in the Key Vault secret key, the system retrieves the active certificate that has the latest expiration date.

![Key Vault parameters](image)

NOTE
The Key Vault storage functionality has been extended so that it includes caching of certificates. The following configuration is recommended:

- Specify a secret version in the Key Vault certificate secret.
- After you upload a new version of the existing certificate to the Key Vault storage, update the `<SecretVersion>` attribute in the Key Vault certificate secret field.

Use the Validate function to verify that you’ve correctly defined the reference to the certificate, and that the certificate is valid.
In Microsoft Dynamics 365 Finance and Dynamics 365 Supply Chain Management, cleanup routines are available in various modules. This topic provides an overview of the routines that are currently available. The information is organized by module.

### System administration

<table>
<thead>
<tr>
<th>PATH</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>System administration &gt; Periodic tasks &gt; Notification clean up</td>
<td>This cleanup routine is used to periodically delete records from the EventInbox and EventInboxData tables. <strong>Recommendation:</strong> If you don't use alert functionality, turn off the alert from the batch job.</td>
</tr>
<tr>
<td>System administration &gt; Periodic tasks &gt; Batch job history clean-up</td>
<td>This regular version of the <strong>batch job history cleanup</strong> routine lets you quickly clean all history entries that are older than a specified number of days. Any entry that was created earlier will be deleted from the BatchJobHistory table, and also from linked tables that have related records (BatchHistory and BatchConstraintsHistory). This version has improved performance optimization, because it doesn't have to run any filtering.</td>
</tr>
<tr>
<td>System administration &gt; Periodic tasks &gt; Batch job history clean-up (custom)</td>
<td>This <strong>custom batch job history cleanup</strong> routine should be used only when specific entries must be deleted. You can clean up selected types of batch job history records, based on criteria such as status, job description, company, or user. You can add other criteria by using the <strong>Filter</strong> button.</td>
</tr>
<tr>
<td>System administration &gt; Inquiries &gt; Database &gt; Database Log &gt; Clean up log</td>
<td>This cleanup routine lets you delete database logs as you require. You can delete logs for specific tables, delete specific types of database logs, or delete logs based on the date and time when they were created. <strong>Note:</strong> Records that have been electronically signed can't be deleted from logs.</td>
</tr>
</tbody>
</table>

### Data management

**IMPORTANT**

These cleanup routines should be run only after the business has done detailed analysis and confirmed that the data is no longer required.

Always test each cleanup routine in a test environment before you run it in a production environment.
In the **Data management** workspace, select **Job history cleanup**.

This cleanup routine is available in Platform update 29 and later. To use it, you must turn on the **Execution history cleanup** feature in Feature management. In **Data management**, this routine must be used to schedule a periodic cleanup of the execution history. It replaces the earlier Staging cleanup routine, which is now obsolete (deprecated).

The following tables will be cleaned up:
- All staging tables
- DMFSTAGINGVALIDATIONLOG
- DMFSTAGINGEXECUTIONERRORS
- DMFSTAGINGLOGDETAIL
- DMFSTAGINGLOG
- DMFDEFINITIONGROUPEXECUTIONHISTORY
- DMFEXECUTION
- DMFDEFINITIONGROUPEXECUTION

In the **Data management** workspace, select the **Staging cleanup** tile.

This cleanup routine should no longer be used, because it’s obsolete. Instead, use the **Job history cleanup** routine.

### General ledger

<table>
<thead>
<tr>
<th>PATH</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| **General ledger > Periodic tasks > Clean up ledger journals** | This cleanup routine deletes General ledger, Accounts receivable, and Accounts payable journals that have been posted. When you delete a posted ledger journal, all information that is related to the original transaction is removed.  

**Note:** You should delete this information only if you’re sure that you won’t have to reverse the ledger journal transactions. |

### Sales and marketing

<table>
<thead>
<tr>
<th>PATH</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sales and marketing &gt; Periodic tasks &gt; Clean up &gt; Delete sales orders</strong></td>
<td>This cleanup routine deletes selected sales orders.</td>
</tr>
<tr>
<td><strong>Sales and marketing &gt; Periodic tasks &gt; Clean up &gt; Delete quotations</strong></td>
<td>This cleanup routine deletes selected quotations.</td>
</tr>
<tr>
<td><strong>Sales and marketing &gt; Periodic tasks &gt; Clean up &gt; Delete return orders</strong></td>
<td>This cleanup routine deletes selected return orders.</td>
</tr>
<tr>
<td><strong>Sales and marketing &gt; Periodic tasks &gt; Clean up &gt; Sales update history cleanup</strong></td>
<td>This cleanup routine deletes old update history transactions. All updates of confirmations, picking lists, packing slips, and invoices generate update history transactions. You can view these transactions on the <strong>History on update</strong> page.</td>
</tr>
</tbody>
</table>
### Procurement and sourcing

<table>
<thead>
<tr>
<th>PATH</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Procurement and sourcing &gt; Periodic tasks &gt; Clean up &gt; Purchase update history cleanup</td>
<td>This cleanup routine is used to delete all updates of confirmations, picking lists, product receipts, and invoices that generate update history transactions.</td>
</tr>
<tr>
<td>Procurement and sourcing &gt; Periodic tasks &gt; Clean up &gt; Delete requests for quotations</td>
<td>This cleanup routine is used to delete requests for quotation (RFQs) and RFQ replies. The corresponding RFQ journals aren't deleted but remain in the system.</td>
</tr>
<tr>
<td>Procurement and sourcing &gt; Periodic tasks &gt; Clean up &gt; Draft consignment replenishment order journal cleanup</td>
<td>This cleanup routine is used to clean up draft consignment replenishment order journals.</td>
</tr>
</tbody>
</table>

### Warehouse management

<table>
<thead>
<tr>
<th>PATH</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warehouse management &gt; Periodic tasks &gt; Clean up &gt; Work creation history purge</td>
<td>This cleanup routine is used to delete work creation history records from the WHSWorkCreateHistory table. In the dialog box, you specify the number of days to keep the history.</td>
</tr>
<tr>
<td>Warehouse management &gt; Periodic tasks &gt; Clean up &gt; Containerization history purge</td>
<td>This cleanup routine is used to delete containerization history from the WHSContainerizationHistory table. In the dialog box, you specify the number of days to keep the history.</td>
</tr>
<tr>
<td>Warehouse management &gt; Periodic tasks &gt; Clean up &gt; Wave batch cleanup</td>
<td>This cleanup routine is used to clean up batch job history records that are related to the wave processing batch group.</td>
</tr>
<tr>
<td>Warehouse management &gt; Periodic tasks &gt; Clean up &gt; Cycle count plan cleanup</td>
<td>This cleanup routine is used to clean up batch job history records that are related to cycle count plan configurations.</td>
</tr>
<tr>
<td>Warehouse management &gt; Periodic tasks &gt; Clean up &gt; Mobile device activity log cleanup</td>
<td>This cleanup routine is used to delete mobile device activity log records from the WHSMobileDeviceActivityLog table. In the dialog box, you specify the number of days to keep the history.</td>
</tr>
<tr>
<td>Warehouse management &gt; Periodic tasks &gt; Clean up &gt; Work user session log cleanup</td>
<td>This cleanup routine is used to delete work user session records from the WHSWorkUserSessionLog table. In the dialog box, you specify the number of hours to keep records.</td>
</tr>
</tbody>
</table>

### Inventory management
<table>
<thead>
<tr>
<th>PATH</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inventory management &gt; Periodic tasks &gt; Clean up &gt; Calculation of location load</td>
<td>The WMSLocationLoad table is used to track the weight and volume of items and pallets. The Summation of load adjustments job can be run to reduce the number of records in the WMSLocationLoad table and help improve performance.</td>
</tr>
<tr>
<td>Inventory management &gt; Periodic tasks &gt; Clean up &gt; Inventory journals cleanup</td>
<td>This cleanup routine is used to delete posted inventory journals.</td>
</tr>
<tr>
<td>Inventory management &gt; Periodic tasks &gt; Clean up &gt; Inventory settlements cleanup</td>
<td>This cleanup routine is used to group closed inventory transactions or delete canceled inventory settlements. By cleaning up closed or deleted inventory settlements, you can help free up system resources. Don't group or delete inventory settlements that are too close to the current date or fiscal year, because part of the transaction information for the settlements will be lost. Closed inventory transactions can't be changed after they have been grouped, because the transaction information for the settlements will be lost. If canceled inventory settlements are deleted, they can't be reconciled with finance transactions.</td>
</tr>
<tr>
<td>Inventory management &gt; Periodic tasks &gt; Clean up &gt; Inventory dimensions cleanup</td>
<td>This cleanup routine is used to maintain the InventDim table. This batch process deletes all existing inventory dimensions that are defined but not used in the current company. All unused inventory dimensions are permanently deleted. No alert or database log is created during this process. This cleanup routine verifies if each InventDim record is being used in not only purchase order lines or sales order lines, but also inventory transactions or on-hand inventory records. If a reference exists to InventDim, it is checked. If it is not used, it will be deleted. If the same combination of dimensions is used later, Dynamics 365 Finance and Dynamics 365 Supply Chain Management will create a new InventDim record with a new InventDimId and use this instead.</td>
</tr>
</tbody>
</table>
| Inventory management > Periodic tasks > Clean up > Dimension inconsistency cleanup | This cleanup routine is used to resolve dimension inconsistencies on inventory transactions that have been financially updated and closed. Inconsistencies might be introduced if the multisite functionality was activated during or before the upgrade process. Use this routine only to clean up the transactions that were closed before the multisite functionality was activated.  

**Note:** Don't use this routine periodically. |
<table>
<thead>
<tr>
<th>PATH</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inventory management &gt; Periodic tasks &gt; Clean up &gt; On-hand entries cleanup</td>
<td>This cleanup routine is used to delete closed and unused entries for on-hand inventory that is assigned to one or more tracking dimensions. Closed transactions contain a value of 0 (zero) for all quantities and cost values, and they are marked as closed. By deleting these transactions, you can help improve the performance of queries for on-hand inventory. Transactions won't be deleted for on-hand inventory that isn't assigned to tracking dimensions.</td>
</tr>
<tr>
<td>Inventory management &gt; Periodic tasks &gt; Clean up &gt; Warehouse management on-hand entries cleanup</td>
<td>This cleanup routine deletes records in the InventSum and WHSInventReserve tables. These tables are used to store on-hand information for items that are enabled for warehouse management processing (that is, WHS items). By cleaning up these records, you can significantly improve the on-hand calculations.</td>
</tr>
<tr>
<td>PATH</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>Inventory management &gt; Periodic tasks &gt; Clean up &gt; On-hand entries aggregation by financial dimensions</td>
<td>Use This cleanup routine as a tool to aggregate InventSum rows that have 0 (zero) quantities. This routine basically extends the previously mentioned routine by also cleaning up records where the <strong>Closed</strong> field is set to <strong>True</strong>. Basically, this routine is needed to handle scenarios where there are no more quantities in the InventSum table for a combination of inventory dimensions, but there is still a value. Although these values will disappear in some cases, the current design occasionally allows values to remain. For example, if you use batch numbers, each batch number (and the combined site, warehouse, and so on) creates a new record in the InventSum table. When the batch number is sold, you will see that quantity fields are set to 0 (zero). In most cases, the <strong>Financial cost amount</strong> and <strong>Physical cost amount</strong> fields are also set to 0 (zero). However, in standard cost revaluation and other scenarios, the field might still show some amount. This behavior is valid, and it reflects the way that Finance and Supply Chain Management handle the costs at the financial inventory level (for example, the site level). In Finance and Supply Chain Management, inventory value is determined by records in the InventSum table. In some cases, when inventory values in the past are reported, it’s determined by inventory transactions (the InventTrans table). Therefore, in the previously described scenario, when you run inventory value reports, Finance and Supply Chain Management initially look at the InventSum table, aggregate all records to the site level, and report the value for the item per site. The data from the individual records at batch number level are never used. Therefore, this routine goes through all InventSum records, finds the records where there is no more quantity (that is, <strong>No open quantities</strong> field is set to <strong>True</strong>). Because there is no reason to keep these records, Finance and Supply Chain Management find the InventSum record for the same item that has the same site, they copy the values from the batch number level to the site level, and they delete the record. Then, when you run inventory value reports, Finance and Supply Chain Management still find the same correct values. Therefore, this routine reduces number of InventSum records, significantly in some cases, and can have a positive impact on the performance of any function that queries that table.</td>
</tr>
<tr>
<td>Inventory management &gt; Periodic tasks &gt; Clean up &gt; Cost calculation details</td>
<td>This cleanup routine is used to clean up cost calculation details.</td>
</tr>
</tbody>
</table>

**Production control**
<table>
<thead>
<tr>
<th>PATH</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Production control &gt; Periodic tasks &gt; Clean up &gt; Production</td>
<td>This cleanup routine is used to delete unused journals.</td>
</tr>
<tr>
<td>journals cleanup</td>
<td></td>
</tr>
<tr>
<td>Production control &gt; Periodic tasks &gt; Clean up &gt; Production</td>
<td>This cleanup routine is used to delete production orders that are ended.</td>
</tr>
<tr>
<td>orders cleanup</td>
<td></td>
</tr>
<tr>
<td>Production control &gt; Periodic tasks &gt; Clean up &gt; Clean up</td>
<td>We recommend that you periodically clean up registrations. This cleanup routine deletes only data that has been processed.</td>
</tr>
<tr>
<td>registrations</td>
<td><strong>Note:</strong> Make sure that you don't delete registrations that might be required later for documentation purposes.</td>
</tr>
<tr>
<td>Production control &gt; Periodic tasks &gt; Clean up &gt; Archive</td>
<td>This cleanup routine is used to remove future registrations from the raw registrations table.</td>
</tr>
<tr>
<td>future registrations</td>
<td></td>
</tr>
</tbody>
</table>

**Master planning**

<table>
<thead>
<tr>
<th>PATH</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master planning &gt; Master planning &gt; Maintain plans &gt; Plan</td>
<td>Usually, this cleanup is done automatically. However, automatic cleanup sometimes malfunctions, and orphan data remains in the system. This orphan data slows down queries and causes the database size to grow. We recommend that you do a preventive run one time per month, when master resource planning (MRP) isn't running.</td>
</tr>
<tr>
<td>version cleanup</td>
<td></td>
</tr>
</tbody>
</table>
Upgrades, updates, and hotfixes can include moving to new product versions, code migration and upgrade, moving to an update, or deploying a hotfix.

The processes for each type of upgrade are similar, but different enough that we think that you should review the topics for a specific task before you begin.

Upgrade from Microsoft Dynamics AX 2012 to Finance and Operations

To get started, review the following topics:

- Upgrade from AX 2012 to Finance and Operations
- Prepare to migrate code to Finance and Operations

Migration from Microsoft Dynamics AX 2009 to Finance and Operations

This Tech Talk video provides an introduction to migration from AX 2009 to Finance and Operations: Dynamics 365 for Operations – Tech Talk: Migration tools.

Upgrade from a previous version of Finance and Operations

The steps for applying updates and upgrading differ between cloud and on-premises implementations.

Cloud

If you are upgrading a cloud version of Finance and Operations, review the following topics:

- Process for moving to the latest update of Finance and Operations
- Apply the latest platform update to environments
- Download updates from Lifecycle Services (LCS)

On-premises

If you are applying updates to an on-premises version of Finance and Operations, review the following topic:

- Apply updates to on-premises deployments
- Redeploy on-premises environments

Hotfixes

- Download updates from Lifecycle Services (LCS)
- Apply updates to cloud environments
- Install metadata hotfixes in development environments
- Patch SQL Server Reporting Services (SSRS) in one-box environments
- Update the Visual Studio development tools

This Tech Talk video provides an introduction servicing (applying code updates, requesting sandbox database

For more information, see:

- Refresh database
- Set up technical support for Finance and Operations apps
Upgrade from AX 2012 to Finance and Operations
11/24/2021 • 13 minutes to read • Edit Online

IMPORTANT
Upgrade is currently only supported from either Dynamics AX 2012 R2 or Dynamics AX 2012 R3. For each release, please update to the latest available cumulative update before upgrading to latest Finance and Operations application release.

Finance and Operations apps provide an upgrade path that customers who currently run Microsoft Dynamics AX 2012 can use to move their data and code to Finance and Operations apps. Currently, upgrades from Dynamics AX 2012 R3 and AX 2012 R2 are supported. The upgrade process is built on the following elements:

- Tools to help you bring forward existing custom application code from AX 2012.
- A data upgrade process that you can use to bring your database forward. Therefore, you can upgrade your full transactional history.

IMPORTANT
Dynamics AX 2012 implementations that are running some deprecated features cannot currently be upgraded. For example, upgrade is not possible from systems that are using either virtual companies or data partitions. If you aren’t sure whether your system can be upgraded, run the Upgrade analyzer tool.

Start your cloud migration journey with a no-charge, no-obligation migration assessment though the Dynamics 365 Migration Program.

Overview
The overall upgrade process can be visualized as three overarching phases: Analyze, Execute, and Validate.

The following diagram shows the end-to-end upgrade process, and the activities that we consider part of each phase.

To learn the most important elements and best practices for successfully upgrading your Dynamics AX 2012 solution to Finance and Operations apps, see Upgrade Dynamics AX 2012 to Finance and Operations apps.

Analyze
The activities in the Analyze phase help you estimate the effort that is required for the upgrade. They also help
you prepare a project plan. These activities can be done before you buy Finance and Operations. They will help you make an informed purchase decision by providing a data point about the effort and resources that you will require.

**Sign up for a preview subscription**
To sign up for a preview subscription, see Sign up for preview subscriptions.

**Append the upgrade methodology**
In your new LCS project, append the project methodology with Upgrade AX 2012 to Dynamics 365 for Finance and Operations. This methodology is specifically for AX 2012 customers who are upgrading. It describes the three phases in detail and provides links to all the supporting documentation about the process.

**Run the upgrade analyzer**
The upgrade analyzer tool runs against your AX 2012 environment and identifies tasks that you should do to prepare the AX 2012 environment, to help make the upgrade experience smoother and less expensive:

- **Data cleanup** – This process helps you identify data that you can remove without causing loss of functionality. The tool identifies various types of data that you can reduce by running a cleanup process. For each type of data, an explanation is given about the impact of the cleanup. You then decide whether to run the cleanup process. Part of the cost of your subscription is based on database size. Therefore, by reducing the size, you reduce that component of the subscription cost and also help reduce the time that is required for the upgrade go-live process. A smaller database helps guarantee a faster upgrade.

- **SQL configuration** – This process reviews the SQL configuration and recommends optimizations. By making sure that SQL performs optimally, this process helps reduce the time that is required for the upgrade go-live process.

- **Deprecated features** – This process identifies features that you're currently using, but that aren't available in Finance and Operations. Therefore, the process helps you discover gaps in functionality early. It also provides suggestions for alternatives.

Additionally, as part of this step, you must install a pre-upgrade checklist in your AX 2012 environment. You can use this checklist to enter data that will be required for the upgrade procedure. For example, in one pre-upgrade checklist task, you provide the Microsoft Azure Active Directory (Azure AD) sign-in information for each current AX 2012 user, so that each user will be able to sign in to Finance and Operations.

- If upgrading from AX 2012 R3, install KB 4035163.
- If upgrading from AX 2012 R2, install KB 4048614.

The output of the upgrade analyzer tool becomes the workstream in the upgrade project plan for your AX 2012 system administrators. For more information, see Upgrade from AX 2012 - Plan by using the Upgrade analyzer tool.

**Run the Code upgrade estimation tools**
This step takes your code from AX 2012, converts it to the new format, and provides feedback about conflicts that a developer must resolve later. This step forms the basis for the estimate of the cost of your code upgrade.

To complete this step, you must export your code from AX 2012 as a model store export and upload it to the LCS Code upgrade tool. The Code upgrade tool will produce an upgraded version of your code and a report about the remaining conflicts that must be resolved. Your developer can then review both the upgraded code and the report to determine the effort that will be required in order to upgrade your code base.

The output of this step represents the workstream in the upgrade project plan for your Microsoft Dynamics AX developers.

For more information, see Upgrade from AX 2012 - Estimate effort by using the Code upgrade service.

**Deploy a demo environment**
Demo environments are default environments that contain demonstration data (not your own data) and standard code (no customizations). We recommend that you deploy a demo environment to evaluate new features, and to perform a basic fit gap analysis of standard processes that are used in AX 2012 but that might have changed in Finance and Operations. You can either deploy these demo environments in Azure or downloaded them as a virtual machine (VM) that you run on your own hardware. If you deploy them in Azure, you must provide your Azure subscription, because you’re still using a public preview project and haven’t yet purchased a subscription.

The output of this step represents the workstream in the upgrade project plan for your functional users or business users.

For more information, see Upgrade from AX 2012 - Deploy a demo environment for analysis

Create a project plan

A template for a project plan is provided in the upgrade methodology. In this step, the output from the previous steps of the Analyze phase is used to fill the project plan for the upgrade project. The project plan will also contain all testing details: data upgrade testing, cutover testing (mock cutover), the functional test pass iterations, and details about the various resource assignments for those tasks.

At this stage, the project plan provides a data point that can help you understand the time and cost of an upgrade.

Execute

During the Execute phase, you work through the tasks that you planned during the Analyze phase. To move to the Execute phase, you must purchase Finance and Operations apps, and you must have available resources that can work on the upgrade.

Switch to the LCS implementation project

The public preview project that you used for the Analyze phase has served its purpose. You can now discard it. For the remaining steps, you require only the project plan that you created in the final step of the Analyze phase.

When you purchase a Finance and Operations subscription, you will receive details about how to sign up for a new LCS project. This project is known as an implementation project and will be the new permanent LCS project for your tenant, for as long as you have that subscription. This project differs from the public preview project in that it's managed by Microsoft. Therefore, this project has these characteristics:

- This project supports deployment of Sandbox and Production type environments.
- The Sandbox and Production type environments are maintained and patched by the Microsoft team, not your staff.
- All DevTest or Demo environments must still be deployed on an Azure subscription. It is recommended to redeploy these environments from your LCS Implementation project so that they are all contained in the same project as your sandbox and production environments.

Identify the project as an AX 2012 upgrade

When you first sign in to your LCS implementation project, you're guided through the Project Onboarding wizard. You can always visit the Project Onboarding wizard later using the navigation menu next to Project Settings in your project.

In the Project Onboarding wizard, in the Project Scope section, you can use the Legacy System field to identify the project as an AX 2012 upgrade. It’s crucial that you identify the project in this way.

Perform the AX 2012 preparation tasks

Complete the tasks that the upgrade analyzer tool discovered, and that are documented in your upgrade project plan. Your Microsoft Dynamics AX system administrator and database administrator (DBA) must complete these
Perform code upgrade
Complete the tasks that were planned during the code upgrade estimation step of the Analyze phase. Your developers must run these tasks.

From this point onward, code changes in AX 2012 should be frozen. Only emergency code changes should be allowed in AX 2012. If a change is made, it must be ported manually to the new code base.

Develop new code
Complete the tasks from the fit gap analysis that was performed during the “Deploy a demo environment” step of the Analyze phase. These tasks will probably be a mixture of functional tasks that define the configuration and development tasks for customizations that are related to new features that are being taken up.

Data upgrade (development environment)
After your code upgrade tasks are completed, you can upgrade your database for the first time. This first upgrade occurs in a development environment, so that you can more easily remediate or debug any issues that are found at this stage. In a development environment, an issue can be debugged immediately, code can be adjusted, and the upgrade can be rerun within minutes. Sandbox environments don’t offer this agility, and a minimum of several hours will be required in order to debug and remediate issues, update code, deploy the updated code, and rerun the upgrade.

The following illustration shows the process. Just back up the AX 2012 database, upload it to Azure, restore it to the Finance and Operations environment, and then run the data upgrade.

Data upgrade is done through a special type of deployable package. The same mechanism is used to deploy new code from one environment to another environment.

The underlying framework that is used to convert the data in the database during this process is largely the same as the upgrade framework in AX 2012 that is based on X++ batch jobs that run ReleaseUpdatexxx classes.

For details, see Upgrade from AX 2012 - Data upgrade in development environments.

NOTE
If you are using Commerce functionality and in-store components as part of the AX 2012 R3 upgrade, we recommend that you review the Dynamics 365 Commerce Phased rollout (N-1) installation, configuration, and cutover guide. For development environments, the user will need to initialize retail parameters, reinitialize the CDX schedule, and then after applying the data upgrade package, the latest quality updates and channel extensions will need to be applied to the environment.

Data upgrade (sandbox environments)
When data upgrade in a development environment completes, you must perform data upgrade in a sandbox environment. For more information about sandbox deployment, see Self-service deployment overview. The sandbox environment is the environment where business users and functional team members can test business processes by using the upgraded AX 2012 data and code.

The following illustration shows the process for running data upgrade in a sandbox environment. The difference
here is that the **AX 2012 Database Upgrade Toolkit for Dynamics 365** is used instead of a traditional SQL backup. This toolkit is required to move your AX 2012 data to Azure SQL Database (using SQL Transaction Replication) as well as to run the data upgrade. In this case, your source is AX 2012 database and target is the Finance and Operations sandbox environment.

For more information, see [Upgrade from AX 2012 - Data upgrade in self-service environments](#).

---

**NOTE**

If you are using Commerce functionality and in-store components as part of the AX 2012 R3 upgrade, we recommend that you review the Dynamics 365 Commerce Phased rollout (N-1) installation, configuration, and cutover guide. For sandbox and development environments, the user will need to initialize retail parameters, reinitialize the CDX schedule, and then reinitialize the CSU after data upgrade. For more information about reinitializing the CSU, see [Initialize Commerce Scale Unit (cloud)](#).

---

**Validate**

When you enter the Validate phase, you will have available environments that include your upgraded custom code and your upgraded data. This phase describes the process of validating and testing that the upgraded environment works as desired. It also describes the process of preparing for go-live.

**Perform cutover testing and create a cutover plan**

The term **cutover** is used here to describe the final process of putting the new system live. This process consists of the tasks that occur after AX 2012 is turned off and before Finance and Operations is turned on.

The goal of the testing, or **mock cutover** is to practice the cutover process. In this way, you can help guarantee that everyone who is involved in the actual cutover to go-live will have a smooth experience.

There are two main workstreams:

- **Technical workstream** – This workstream is the process of running the data upgrade. Your business will enforce a limit on the amount of downtime that is allowed. During this downtime, neither product database will be available. The technical workstream might have to performance-tune its data upgrade procedure to meet the business’s downtime limit.

- **Functional workstream** – After data upgrade, several configuration tasks will be required in the Finance and Operations environment. All these tasks must be documented and quantified, and a resource must be assigned to them, because they must fit together with the technical tasks within the business’s downtime limit.

For additional details, see:

- [Upgrade from AX 2012 - Post-upgrade tasks](#)
- [Upgrade from AX 2012 - Cutover testing (Mock cutover)](#)

**Functional test pass**

Complete a full functional test pass of all business processes. This test pass will be an extensive retest of all business processes that involve Finance and Operations. These business processes include both old processes that were brought forward from AX 2012 and new processes that involve new features that were taken up for the first time in Finance and Operations.

Depending on code quality, issue remediation and retesting might require several iterations of the functional test pass. When an issue is fixed, be sure to retest all processes that are involved, to help guarantee that the
downstream or upstream process isn't affected by the change.

For details, see Upgrade from AX 2012 - Functional test passes.

**Pre-go-live checklist**

The pre-go-live checklist is a recommended procedure that can help reduce the chance of errors during the final cutover to go-live. One week before go-live is due, stop configuration changes in AX 2012 (that is, under <module>\Setup). This restriction on configuration changes is merely procedural. The Microsoft Dynamics AX system administrators just agree to put changes of this type on hold at this point.

We recommend that you also freeze code changes in the Finance and Operations code base. No further changes should be allowed unless they have been evaluated and have been shown not to block go-live.

After the configuration restriction and code freeze are in place, data upgrade should be run for the last time before cutover. In this way, you can make sure that everything still works as expected.

For more information, see Validate: Prepare for go live.

**Go live**

After you have successfully completed upgrade testing in a Standard or Premier Acceptance Test environment (Sandbox Tier 2 or higher), and you have also completed a successful test cutover, the moment has arrived to upgrade your production environment and go live.

*Cutover* is the term that we use for the final process of getting a new system live. This cutover process consists of the tasks that occur after Dynamics AX 2012 is turned off but before Finance and Operations is turned on.

For details, see Upgrade from AX 2012 - Cutover process (Go live).

**Supported upgrade paths**

Upgrade to the cloud version of Finance and Operations apps is supported from AX 2012 R2 and AX 2012 R3.

Upgrade from Dynamics AX 2012 RTM isn't currently supported. Upgrade to the on-premises version is supported, as documented in Data upgrade process for AX 2012 to Dynamics 365 Finance + Operations (on-premises).
This topic explains how to use the Upgrade analyzer tool to plan your upgrade from Microsoft Dynamics AX 2012. This tool is run against an AX 2012 environment and identifies data that you should clean up in AX 2012 to help reduce the subscription cost for Finance and Operations. The tool also suggests SQL configuration optimizations that can help speed up the upgrade processes. Additionally, the tool warns you if any features that you use in AX 2012 are obsolete in the current version. Therefore, you can plan ways to replace or work around those features.

Upgrade analyzer gathers data from your AX 2012 environment as part of the regular System diagnostic service in Dynamics Lifecycle Services (LCS). For an overview of the System diagnostic service, and for information about how data is collected and pushed back into the cloud so that you can consume it through LCS, see System diagnostics in Lifecycle Services (LCS).

You can view the results of the System diagnostic service in a Microsoft Power BI report in LCS. The report presents a list of tasks that you should complete in the AX 2012 environment.

To access the Upgrade analyzer report, go to https://diag.lcs.dynamics.com/UpgradeAnalysisReport/Report/"ProjectID" (Replace "ProjectID" with your current project ID, which is an integer that can be found in the URL of your current LCS project).

The following illustration shows an overview of the procedure for using Upgrade analyzer.

If you already use the System diagnostic service in your AX 2012 environment, you must configure a new instance of the service on a machine that differs from the existing machine.

For information about how to configure the System diagnostic service in your AX 2012 environment, see Install and run System diagnostics.

Within a few minutes after you configure the System diagnostic service, the AX 2012 environment will appear in your LCS project.
This topic explains how to use the Code upgrade service in Microsoft Dynamics Lifecycle Services (LCS) to help estimate the tasks and effort that are required in order to upgrade a code base from Microsoft Dynamics AX 2012 Finance and Operations.

The Code upgrade service converts an export of your AX 2012 model store to the correct format. However, the new version of your code won’t be fully functional until a developer resolves any issues that the service identifies but can’t resolve itself.

The Code upgrade service performs these actions:

- Directly resolve some types of conflict issues.
- For other issues, log Microsoft Azure DevOps tasks.
- Create a version of your code in the correct format, and check the new version into a new branch of your Azure DevOps project.

In the Analyze phase, we use the report to help estimate the effort that is required in order to complete code conversion activities.

The following illustration shows an overview of the process for configuring the Code upgrade service.

For information about how to configure the Code upgrade service, see Configure the code upgrade service in Lifecycle Services (LCS).

The output of the Code upgrade service is designed to be consumed by a developer. This output will help the developer estimate the effort that is required in order to complete the code upgrade tasks. To form an estimate, the developer must review the tasks that the service generates in Azure DevOps and the new version of the code that the service generates.
This topic explains why and how you should deploy a demo environment during the Analyze phase of your project for upgrading from Microsoft Dynamics AX 2012 to Finance and Operations.

By deploying a demo Finance and Operations environment, you gain hands-on experience with the program and can explore new features that you might be interested in. You also have an opportunity to validate differences in the program for your business processes, so that you can identify potential gaps or confirm that no gaps exist. At this stage in your upgrade project, your data and code won’t be available in the environment. Therefore, you will have limited ability to validate that everything conforms to your business process. However, this step is the first step in that work stream.

If you haven’t yet purchased licenses, and you’re using a free trial, you can follow the steps in Deploy a demo environment to deploy a demo environment to a Microsoft Azure subscription that you bring yourself.

If you’ve already purchased licenses, you received a link to configure a special type of project in Microsoft Dynamics Lifecycle Services (LCS): an implementation project. The implementation project will let you deploy a dev/test environment and a sandbox environment. For more information about this type of environment deployment, see Upgrade from AX 2012 - Data upgrade in sandbox environments.
This is an exciting moment in the upgrade project. The output of this task provides the first upgraded dataset from Microsoft Dynamics AX 2012 to the latest Finance and Operations development environment.

Before you run this process in a shared sandbox environment, we recommend that you run it in a development environment. There are two reasons for this approach:

- It provides local data that developers can write and test their custom data upgrade scripts against.
- It helps reduce the overall time that is spent on iterations of the data upgrade process. In a development environment, an issue can be debugged immediately, code can be adjusted, and the upgrade can be rerun within minutes. However, larger sandbox environments don't allow for this level of agility. In those environments, a minimum of several hours will be required to debug and remediate issues, update code, deploy the updated code, and rerun the upgrade.

We strongly recommend that you run the Upgrade analyzer and respond to the issues it identifies before running data upgrade - this will help ensure that your data upgrade is quicker and easier.

End-to-end data upgrade process

Back up your AX 2012 database
To back up your AX 2012 database, use the standard Microsoft SQL Server process to produce a BAK file. If you use the compression option when you create the backup, the file size will be smaller, and less time is required in order upload it to and download it from Microsoft Azure Storage.

Upload the backup to Azure Storage
If your developer environment is hosted as a VM locally or in Azure, you will need to transfer the 2012 database backup to it. With a local VM you may be able to transfer the file directly across the network (if you have configured the virtual network to allow that) but for an Azure hosted VM we recommend that you upload your backup to Azure Storage (using your own secure file transfer service or SFTP is also a valid option). You would need to provide your own Azure storage account for this. There are free tools to help you to move files between Azure storage, from a command line you can use Azcopy, or for a GUI experience you can use Microsoft Azure storage explorer. Use one of these tools to first upload the backup from your on-premises environment to Azure storage and then on your download it on your development environment.

Download and restore the backup to the customer-managed development environment
When you restore the backup to the new development environment, don’t overwrite the existing AXDB database. Instead, restore the AX 2012 database next to the original databases. You might also consider using drive D for the data and log files, to help improve performance. However, there is a potential downside to using drive D. If the underlying virtual machine (VM) is deallocated in Azure and then reallocated, drive D will be wiped. In practice, this scenario rarely occurs. Therefore, you might find that the risk is acceptable. To learn more about how to use drive D, see [Understanding the temporary drive on Windows Azure Virtual Machines](#).

To speed up the database restore process, you can change the SQL Server service account to `axlocaladmin`. The restore process can then use instant file initialization. For more information, see [Database Instant File Initialization](#).

After the database is restored, stop the following services:

- World wide web publishing service
- Dynamics 365 for Finance and Operations Batch Management service
- Management Reporter 2012 Process service
- Microsoft Dynamics Lifecycle Services Diagnostic Service
- Data Import / Export service

Next, rename the original AXDB database `AXDB_orig`. This database might be useful as reference later, when you develop code.

```
ALTER DATABASE AXDB SET SINGLE_USER WITH ROLLBACK IMMEDIATE
GO
ALTER DATABASE AXDB MODIFY NAME = AXDB_Orig
GO
ALTER DATABASE AXDB_Orig SET MULTI_USER
GO
```

Finally, rename the newly restored AX 2012 database `AXDB`.

**Run the data upgrade deployable package**

To get the latest data upgrade deployable package for a target environment that is running the latest update, download the latest binary updates from Microsoft Dynamics Lifecycle Services (LCS) Shared asset library.

1. Sign in to [LCS](#).
2. Select the `Shared asset library` tile.
3. In the `Shared asset` library, under `Select asset type`, select `Software deployable package`.
4. In the list of deployable package files, find the data upgrade package that corresponds to your upgrade. For example, if you're upgrading from AX 2012, the package name starts with AX2012DataUpgrade. Select the package that corresponds to the release you are upgrading to. For example: AX2012DataUpgrade-July2017.

For more information, see [Upgrade data in development or demo environments](#).

**Troubleshooting data upgrade script errors**

There are options that let you resume the data upgrade where it last stopped. You can also record any data upgrade script errors with call stacks to a table in the database. For development scenarios, you can skip failed scripts and continue to run the upgrade.
Recommendation for the first data upgrade run

When you run the data upgrade against your dataset for the first time, and especially when there many customizations or many custom data upgrade scripts, you might find the feature to skip failed scripts useful. By using this feature, you gain visibility into as many errors as possible in one run. Otherwise, only one critical issue is discovered per run. Be aware that, because dependencies exist between scripts, you might receive errors in related child scripts if you skip the parent script. These errors occur only because the parent wasn’t run correctly. They will be resolved when the issue in the parent script is resolved.

For more details, see the main data upgrade topic.
Upgrade from AX 2012 - Pre-upgrade checklist for data upgrade

This topic describes each task in the Microsoft Dynamics AX 2012 checklist that is associated with data upgrade to Finance and Operations.

Installation

Use the pre-upgrade checklist to enter data that will be required for the upgrade procedure.

- If upgrading from AX 2012 R3, install KB 4035163
- If upgrading from AX 2012 R2, install KB 4048614

Prepare model metadata

During data upgrade, one goal is to maintain element IDs between the existing AX 2012 environment and the upgraded Finance and Operations environment. To accomplish this goal, you must bring a copy of the element IDs from the AX 2012 environment into the Finance and Operations environment. AX 2012 stores element IDs in a table that is named ModelElement. This table is in the model database, which is a separate database from the AX 2012 business data database. During an upgrade to Finance and Operations, you must copy the AX 2012 database to Microsoft Azure. This process can be time consuming.

To avoid copying the whole model database to Azure SQL Database, use the following procedure to replicate the ModelElement table in the business data database. Later, during data upgrade runs, the database synchronization process will retrieve the required information from this replicated table and make sure that element IDs are maintained in the upgraded Finance and Operations environment.

1. In the Finance and Operations data upgrade checklist, click Prepare model metadata.
2. When you’re prompted, click Yes.
3. Wait for the copy process to be completed.

If the process is successful, the task is marked as completed.

Prepare security role metadata

Another goal during data upgrade is to preserve security role assignments. This task resembles the previous "Prepare model metadata" task. Security role information that is stored in the AX 2012 model database must be copied to the AX 2012 business data database, so that the information is preserved in the Finance and Operations environment after upgrade. During data upgrade runs, the same security role will be restored in the upgraded Finance and Operations environment.

1. In the Finance and Operations data upgrade checklist, click Prepare security role metadata.
2. When you’re prompted, click Yes.
3. Wait for the copy process to be completed.

If the process is successful, the task is marked as completed.

**Set up user mapping**

In AX 2012, users are authenticated against an on-premises Active Directory server. However, in Finance and Operations, users are authenticated against Azure Active Directory (Azure AD). This task provides a form where you can map existing AX 2012 users to equivalent Azure AD users. The AX 2012 users will then be able to access Finance and Operations.

1. In the Finance and Operations data upgrade checklist, click **Set up user mapping**.

2. The **User info email mapping** form appears. Follow one of these steps to fill in the grid:
   - Import users from AX 2012, and then manually fill in the Azure AD email address:
     a. Click **Import from AX**. The grid is filled with existing users.
     b. For each user, enter the corresponding Azure AD email address, as shown in the following illustration.

   ![User info email mapping](image)

   - Import users from a file. This option is faster. We recommend that you use this option when many users must be updated.
     a. In a comma-separated values (CSV) file, create the mapping between AX 2012 users and Azure AD email addresses. Your IT department can export a similar mapping from your on-premises Active Directory Domain Services (AD DS). The file should have two columns: **UserId** and **EmailAddress**.

     **NOTE**

     The first row in the file is treated as a header row and will be ignored during the import.

     b. After the file is ready, click **Import from file**, browse to the file, and import it.

     The grid should be filled with the mappings that you specified in the file.

If the imported file contains an entry that isn’t valid, an error file is generated.

**Validate baseline version**

Run this task to validate that the current version can be upgraded.

- In the Finance and Operations data upgrade checklist, click **Validate baseline version**.

If the baseline version is one of the supported baseline versions, the task is marked as completed.

**Archive retail salt data**

This task is used to migrate the registry key that RetailSaltUtility uses. This tool is used for some deployments where the customer wants to inject a specific random value into the hash that is used to authenticate channel
users.

- In the Finance and Operations data upgrade checklist, click **Archive retail salt data**.

If the process is successful, the task is marked as completed.
Upgrade from AX 2012 - SQL Transactional Replication

11/24/2021 • 8 minutes to read • Edit Online

**IMPORTANT**
Upgrade is currently only supported from either Dynamics AX 2012 R2 or Dynamics AX 2012 R3. For each release, please update to the latest available cumulative update before upgrading to latest Finance and Operations application release.

This topic shows how to upgrade a large Microsoft Dynamics AX 2012 database to Finance and Operations apps. This process uses SQL Transactional Replication to bring the schema and data from the AX 2012 on-premises database to the sandbox environment.

We strongly recommend that you run the data upgrade process in a development environment before you run it in a shared sandbox environment. This approach will help reduce the overall time that is required for a successful data upgrade. For more information, see Upgrade from AX 2012 - Pre-upgrade checklist for data upgrade.

**Replication setup**

Replication is a set of technologies for copying and distributing data and database objects from one database to another, and then synchronizing between the databases to maintain consistency. This migration and copying happens with the source system online, which means there is no need for Finance and Operations service downtime during the replication process. The Online Database Migration Toolkit uses transactional replication. This is typically used in server-to-server scenarios that require high-throughput to improve scalability and availability.

The Online Database Migration Toolkit can be downloaded from Lifecycle Services (LCS) in Shared asset Library > Model.

**Prerequisites**

The following prerequisites are needed for the Online Database Migration Toolkit.

- The source SQL Server should have the replication feature installed and enabled. To check whether replication is enabled, execute the following SQL script:

```sql
-- If @installed is 0, replication must be added to the SQL Server installation.
USE master;
GO
DECLARE @installed int;
EXEC @installed = sys.sp_MS_replication_installed;
SELECT @installed;
```

If the replication components are not installed, follow the steps in Install SQL Server replication.

- SQL Agent should be running in the source database server.

- SA Authentication: A user should have DB_Owner privilege in the source database and the target database. In the source database, the user should have access to masterDb and sourceDb.
- Update the target firewall by allow-listing the source IP. This can be done via LCS. This only allows for 8 hours of access. After allow-listing, you need to execute the following stored procedure in the target database to have more than 8 hours of access.

```sql
-- Create database-level firewall setting for IP a.b.c.d
EXECUTE sp_set_database_firewall_rule N'AX 2012 Upgrade', 'a.b.c.d', 'a.b.c.d';
```

- To optimize the replication latency/performance, the following are fine-tuned distributors parameters that can be updated in the `params.xml`.
  - MaxBcpThreads
  - NumberOfPublishers
  - Distributor database paths

- Stop the AOS service in the target environment, so that the target database will get replicated more efficiently. Running the AOS in the target may cause slowdown in the replication process. This may cause schema lock or deadlock in the replication process.

- When setting up Distributor: The script creates a database in the source server. Be sure you have enough space. The recommended is minimum to have the size of the source database. In `params.xml`, specify the distributor database path so that the database can be created in the specified path.

- Update `params.xml`

```xml
<?xml version="1.0" encoding="UTF-8"?>
<! -- Database replication parameters for an AX 2012 to Dynamics 365 upgrade -->
<Config>
  <!-- Edit the properties in this section for your source AX 2012 database -->
  <SourceDatabase>
    <Server>SQLSERVERNAME\SQLINSTANCE</Server>
    <Database>MicrosoftDynamicsAX</Database>
    <UserName>ReplicationUser</UserName>
    <Password>********************</Password>
  </SourceDatabase>
  <!-- Edit the properties in this section for your target Dynamics 365 database -->
  <TargetDatabase>
    <Server>dbmigration.database.windows.net</Server>
    <Database>dbms-prod</Database>
    <UserName>axdbadmin</UserName>
    <Password>*******************</Password>
  </TargetDatabase>
  <!-- Edit the properties in this section for your local SQL replication settings -->
  <SQLReplicationSettings>
    <!-- Ensure that you have enough space in the drive/path -->
    <SnapshotWorkingDir>D:\SQLServer\Snapshot</SnapshotWorkingDir>
    <DistributorDBDataFolder>D:\SQLServer\Data</DistributorDBDataFolder>
    <DistributorLogFolder>D:\SQLServer\Data</DistributorLogFolder>
    <!-- Based on the number of cores, you can set this, but the max this value can be is 8. This value should be between 4 to 8 -->
    <MaxBcpThreads>4</MaxBcpThreads>
    <!-- To increase the performance of the replication. This value should be between one and three. This value will be used to create the number of publishers for tables with primary keys. -->
    <NumberOfPublishers>2</NumberOfPublishers>
    <!-- Ignore DB objects xml file. The database objects listed in these files will be not be replicated. -->
    <IgnoreTablesList>Data\ignoretables.xml</IgnoreTablesList>
    <IgnoreFunctionsList>Data\ignorefunctions.xml</IgnoreFunctionsList>
  </SQLReplicationSettings>
</Config>
```

- XML Schema: To ignore selected tables, views, and functions during replication, add this information.
Configuring replication

The **SQLTransactionalReplication** folder has all the Windows PowerShell scripts that are required to configure the SQL transactional replication. These scripts should be executed using the following sequence. Be sure to wait for the process to finish.

1. **Replication_01_DataBaseCleanup.ps1** - Will empty the target database.

2. **Replication_02_Distributor.ps1** - Upon completion, the distributor database will get created in the source database server under the system database.

3. **Replication_03_PublisherTables.ps1** - After the publisher scripts are successfully executed, publication will be created under the replication folder. Note that this will take some time to complete. This creates publishers AXDB_PUB_TABLE_Obj_[*].

   **WARNING**
   
   Wait for data replication to complete before executing cutover scripts. You can check the status in the following ways:
   
   - Replication monitor: On the source server, right-click the Replication folder and select Launch Replication Monitor.
   - Run GetStatus.ps1 script embedded in the replication toolkit. **DataReplicationStatus** must be set to complete for each AXDB_PUB_TABLE_Obj_[*] publication.

4. **Replication_04_PublisherOtherObjects.ps1** - Replicates functions to the target database by creating new publication. This step can be omitted if you don’t want to move functions. Note that this will be completed quickly. This creates publisher AX_PUB_OtherObjects.

5. **CutOver_01_PublisherNoPK.ps1** - This creates two publications to replicate: - Non-primary key tables
   - Locked tables with publication names: AX_PUB_NoPKTable, AXDB_PUB_TABLE_Locked

6. **CutOver_02_PKDeletion_PostReplication.ps1** - This will clean up the temp tables created for tables with no primary keys. Deletes publication AX_PUB_NoPKTable.

7. **CutOver_03_RetrieveAndCreateNoPKConstraints.ps1** - This extracts constraints for the tables with no primary keys from the source and creates them in the target database.

8. **CutOver_04_RemoveReplication.ps1** - After successful replication of the database, you can execute this script to remove replication setup information. If you want to remove the snapshot folder without errors, execute the following stored procedure in the source Db. Otherwise, after execution you will get an error that the system was unable to remove the snapshot folders, which should be removed manually.

```sql
EXEC master.dbo.sp_configure 'show advanced options', 1
RECONFIGURE WITH OVERRIDE
EXEC master.dbo.sp_configure 'xp_cmdshell', 1
RECONFIGURE WITH OVERRIDE
```

The following publications will get created in the source database when setting up the replication.
Find the replication status and get an exception

To find the replication status and get an exception, execute the PowerShell script and wait for it to finish.

**GetStatus.ps1** - When you execute this script, the Replication status table will be listed, along with the schema AgentId, PublicationName, Job, LastSynced, JobStatus, ReplicationStatus, and Comments.

- **AgentId** - Used to fetch the exception details about the job.
- **PublicationName** - The publication names created for replicating the data. You can find the same information in the SQLServerExplorer under Replication folders.
- **Job** - There are two types of jobs: Snapshot and Data Replication.
- **JobStatus** - This will display the statuses of Started, Succeeded, In Progress, Idle, Retrying, or Failed. For snapshot jobs, after the status is Succeeded this will no longer execute. For data replication jobs, the status will continue to change based on the update information in the database.
- **ReplicationStatus** - This applies only to the data replication jobs. Statuses include Waiting, In Progress, and Completed.
- **Comments** - This will continue to change when the JobStatus is InProgress.

**GetException.ps1** - Provide the AgentId to get the exceptions. AgentId can be retrieved from the status.

Find the replication configuration and status via SQL Server Management Studio

To find the replication status configuration and status using SQL Server Management Studio, follow these steps:

- To determine if the replication feature is available and installed on the server, you should see the Replication folder in Object Explorer.
After executing the `Replication_03_PublisherTables.ps1` script, you should be able to see the publisher configured under the Replication folder.

To determine the replication status, right-click the Replication folder and select Launch Replication Monitor.
In the Replication Monitor window, you can see all the publishers that have been created for replication.

Select the Snapshot tab to see the status of the snapshot.

To view the detail log/transaction, double-click the item.
To view the data replication to the target, select the **All Subscription** tab and double-click the subscription for the item.

### Troubleshooting

<table>
<thead>
<tr>
<th>EXCEPTION</th>
<th>SOLUTION/FIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SOLUTION/FIX</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>The subscriptions have been marked as inactive and must be</td>
<td>1) Check the status in the source database using the</td>
</tr>
<tr>
<td>reinitialized. NoSync subscriptions will need to be dropped</td>
<td>following query and update the status to &quot;2&quot; for the specific</td>
</tr>
<tr>
<td>and recreated. (Source: MSSQLServer, Error number: 21074)</td>
<td>publication</td>
</tr>
<tr>
<td></td>
<td>Check the status, you can get the srvname from this output</td>
</tr>
<tr>
<td></td>
<td>query</td>
</tr>
<tr>
<td></td>
<td><code>select * from syssubscriptions WHERE status != 2</code></td>
</tr>
<tr>
<td></td>
<td>Update only if the status != 2</td>
</tr>
<tr>
<td></td>
<td><code>Update syssubscriptions SET status = 2 where srvname = 'your target server name'</code></td>
</tr>
<tr>
<td></td>
<td>2) Check the status in the distributor database with the</td>
</tr>
<tr>
<td></td>
<td>following query and update the status to &quot;2&quot; for the specific</td>
</tr>
<tr>
<td></td>
<td>publication</td>
</tr>
<tr>
<td></td>
<td>To get the publication_id, use this following query and</td>
</tr>
<tr>
<td></td>
<td>match this with your publication name</td>
</tr>
<tr>
<td></td>
<td><code>SELECT * FROM MSpublications</code></td>
</tr>
<tr>
<td></td>
<td>Check the status using the following query</td>
</tr>
<tr>
<td></td>
<td><code>SELECT * FROM MSsubscriptions WHERE status != 2</code></td>
</tr>
<tr>
<td></td>
<td>publication_id = &lt;@publicationId&gt;`</td>
</tr>
<tr>
<td></td>
<td>Update if the status is != 2 for that specific</td>
</tr>
<tr>
<td></td>
<td>publication_id = &lt;@publicationId&gt;`</td>
</tr>
<tr>
<td></td>
<td>Execute this in the source database and sign in with the</td>
</tr>
<tr>
<td></td>
<td>credentials that you used to create the publication</td>
</tr>
<tr>
<td></td>
<td>EXEC <code>sp_changdbowner 'sa'</code></td>
</tr>
<tr>
<td></td>
<td>Execute this in the source database;</td>
</tr>
<tr>
<td></td>
<td>Clean the subscription:</td>
</tr>
<tr>
<td></td>
<td><code>exec sp_subscription_cleanup @publisher = @publisherServer, @publisher_db = @publisherDb, @publication = @publicationName</code></td>
</tr>
<tr>
<td></td>
<td>Drop the subscription:</td>
</tr>
<tr>
<td></td>
<td><code>exec sp_droppshibition @publication = @publicationName, @subscriber = N'all', @article = N'all</code></td>
</tr>
<tr>
<td></td>
<td>Drop the publication:</td>
</tr>
<tr>
<td></td>
<td><code>exec sp_droppublication @publication = @publicationName</code></td>
</tr>
</tbody>
</table>

Error messages:

* The process could not execute 'sp_replicmds' on 'replicationsrv\MSSQLSERVER2016'. (Source: MSSQL_REPL, Error number: MSSQL_REPL20011)
  Get help: http://help/MSSQL_REPL20011

* Cannot execute as the database principal because the principal "dbo" does not exist, this type of principal cannot be impersonated, or you do not have permission. (Source: MSSQLServer, Error number: 15517)
  Get help: http://help/15517

* The process could not execute 'sp_replicmds' on 'replicationsrv\MSSQLSERVER2016'. (Source: MSSQL_REPL, Error number: MSSQL_REPL22037)
  Get help: http://help/MSSQL_REPL22037

To remove/delete a publication

* Execute this stored procedure in the source database;
<table>
<thead>
<tr>
<th>EXCEPTION</th>
<th>SOLUTION/FIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>To remove an article from the publication, see <code>sp_dropsubscription</code> (Transact-SQL)</td>
<td>Execute this stored procedure in the source database:</td>
</tr>
<tr>
<td></td>
<td><code>EXEC sp_dropsubscription</code></td>
</tr>
<tr>
<td></td>
<td><code>@publication = @publication,</code></td>
</tr>
<tr>
<td></td>
<td><code>@article = N'All',</code></td>
</tr>
<tr>
<td></td>
<td><code>@subscriber = @subscriber;</code></td>
</tr>
<tr>
<td>Example:</td>
<td><code>EXEC sp_dropsubscription @publication = N'OtherObjects_sp',</code></td>
</tr>
<tr>
<td></td>
<td><code>@article = N'MaintainShipCarrierRole',</code></td>
</tr>
<tr>
<td></td>
<td><code>@subscriber = N'SPARTAN-SRV-NAM-D365OPSDEV-D5E38124F9F8.DATABASE.WINDOWS.NET';</code></td>
</tr>
</tbody>
</table>
This Microsoft Dynamics AX 2012 data upgrade process is for self-service environments. Complete the sections of this topic in the following order:

1. **Prerequisites**
2. **Data upgrade process** – Run the AX2012DataUpgradeToolKit.exe application to complete the upgrade process.
3. **Reporting section of the application** – Review the reports of the replication validation, replication status, data upgrade status, and rollback data upgrade status.
4. **Tooling section of the application** – This section will help you reset the process parameters and restart any of the processes.
5. **Troubleshooting**

### Prerequisites

1. Download the AX 2012 Database Upgrade Toolkit for Dynamics 365 from Microsoft Dynamics Lifecycle Services (LCS). In the Shared asset Library, select **Model** as the asset type, and then select the model file.

2. Create a self-service environment in LCS. The environment should be in a **Deployed** state. It must be a Microsoft-managed environment. Cloud-hosted, development environments can be used only for the Upgrade from AX 2012 - Data upgrade in development environments procedure.

3. Download and install the .NET Framework version 4.7.1 if it isn't already installed.

4. Make sure that the replication feature is installed and enabled for the source SQL Server instance. To determine whether replication is enabled, run the following SQL script.

   ```sql
   -- If @installed is 0, replication must be added to the SQL Server installation.
   USE master;
   GO
   DECLARE @installed int;
   EXEC @installed = sys.sp_MS_replication_installed;
   SELECT @installed;
   
   If the replication components aren't installed, follow the steps in Install SQL Server replication to install them.

5. Enable and start the SQL Server Agent on the source database server.

   **NOTE**
   
   A user should have the **DB_Owner** privilege in the source database, and should have access to the master database and the source database.
6. **Migration toolkit setup:** If you don’t want some of the source database tables to be replicated in the target database, you can specify them in the IgnoreTables.xml file. Likewise, if you don’t want some of the functions to be replicated, you can specify them in the IgnoreFunctions.xml file.

- **Path of the IgnoreTables.xml file:** `Data\IgnoreTables.xml`
- **Path of the IgnoreFunctions.xml file:** `Data\IgnoreFunctions.xml`

The following examples show how to specify tables and functions in the XML files.

```xml
<?xml version="1.0" encoding="utf-8"?>
<IgnoreTables>
    <Name>
        <Table>NON_AOT_TABLE1</Table>
        <Table>NON_AOT_TABLE2</Table>
        <Table>NON_AOT_TABLE3</Table>
    </Name>
</IgnoreTables>
```

**NOTE**

The tables added to the ignore list should only be tables that do not exist in the Microsoft Dynamics AX 2012 Application Object Tree (AOT). Including tables that exist in the AOT will result in an error during the data upgrade.

```xml
<?xml version="1.0" encoding="utf-8"?>
<IgnoreFunctions>
    <Name>
        <Function>if_WHSInventReserveUnionDelta</Function>
    </Name>
</IgnoreFunctions>
```

**IMPORTANT**

The tables and functions that are specified in these XML files won’t be replicated in the target database, and the same format should be followed.

7. To optimize the replication latency/performance, you can update the following distributor parameters in the `App.config` file:

- **MaxBcpThreads** – By default, this parameter is set to 6. If the machine has fewer than six cores, update the value to the number of cores. The maximum value that you can specify is 8.
- **NumberOfPublishers** – By default, this parameter is set to 2. We recommend that you use this value.

**NOTE**

Do not set up or configure replication during peak times when the system resources/memory usage/IO operations are high. When resources are being used to the max (greater than 90% is already consumed) then the replication may be delayed as the system tries to find available resources. We recommend that you start the replication during off hours, when the system resources are at minimum usage (during off-peak time). Additionally, it is recommended for a go-live cutover that you start the replication the prior weekend.

---

**Data upgrade process**

**Run the AX2012DataMigration.exe application**
Before you begin the replication process, note that the LCS environment will be in a **Deployed** state when it’s created.

1. Run the **AX2012DataMigration.exe** application.
   
   A console window will open, and it prompts you to sign in.

2. Provide the credentials that are used to sign in to LCS.

3. After you’re successfully authenticated, in the console window, provide the **Project-Id** value and then the **Environment-Id** value.

   To validate the given values, you will need to sign in using the credentials that are used to sign in to LCS.

   **NOTE**
   
   You can find the **Project-Id** and **Environment-Id** values on the **Manage environment** page in LCS. You can also find the **Environment-Id** value on the **Environment details** page.

**Complete the data replication and upgrade**

After the validation is successful, the application presents a set of menu options that correspond to the steps in the data upgrade process. To complete the data replication and upgrade, you should perform the steps in the following order.

1. **Data upgrade preparation: Environment setup activity**

   This step prompts you for the following information:

   - **Details of the source database**:
     
     - Source server (in the format `servername\serverinstance`)
     - Source database name
     - User name
     - Password
   
   - **IP address of the source database server** (for the allowlist)
   
   - **Distribution database path** (for example, `D:\SQLServer\Data`)
   
   - **Replication snapshot path** (for example, `D:\SQLServer\Snapshot`)

   **IMPORTANT**

   The specified distribution database and replication snapshot paths should have enough space. We recommend that the amount of space be at least the size of the source database. The paths should be in the local disk of the machine. Avoid using shared paths.

   We recommend that you have a static IP address for the virtual machine (VM) or machine (for the allowlist in step 1). In this way, you help prevent connection issues with the target database.

   This step performs the following actions:

   - It validates the connection to the source database.
   - It validates the version of the AX 2012 database.
   - It authorizes the source IP address.
   - It validates the target databases.

2. **Data upgrade preparation: Prepare the target environment for the data upgrade**
This step changes the state of the LCS environment from **Deployed** to **Ready for replication**.

3. **Replication: Clean-up target database**

   This step performs the following actions:
   
   a. Change the state of the LCS environment from **Ready for replication** to **Replication in progress**.
   
   b. Delete all AX product tables, views, stored procedures, and user-defined functions in the target database.

4. **Replication: Set up distributor**

   This step creates a distribution database under the **System Databases** folder on the source server. This distribution database is used for replication.

5. **Replication: Set up publication for primary key tables**

   This step creates publications for primary key tables under the **Replication** folder on the source server and replicates them in the target database. If any **ignore-table** entries are specified, the specified tables are exempted from replication.

   **Created publishers:** AXDB_PUB_TABLE_Obj_*

   **NOTE**
   
   After this replication configuration step is completed, actual data replication will occur as a SQL job that runs in the background. This job will take some time to be completed. You can view the status of the replication by providing the ‘rs’ option. To learn more about the ‘rs’ option, see the Reporting section of the application section later in this topic.

6. **Replication: Set up publication for other objects (functions)**

   This step creates a publication for other objects (functions) and replicates them in the target database. If you don’t want some of the functions to be replicated, you can specify them in the IgnoreFunctions.xml file.

   **Created publisher:** AX_PUB_OtherObjects

   **NOTE**
   
   The replication will take some time to be completed. You can view the replication status by providing the ‘rs’ option.

   If there are no functions to replicate, the publication won’t be created.

   Don’t move on to next step until the **DataReplicationStatus** property for this step is shown as completed.

7. **Cutover: Set up publication for non-primary key tables**

   This step creates two publications: one that is used to replicate non-primary key tables, and one that is used to replicate locked tables.

   **NOTE**
   
   If there are no locked tables, then publication will not be created.

   **Publication names:** AX_PUB_NoPKTable, AX_PUB_TABLE_LockedTable

   If AX Service acquires a schema lock during creation of the primary key publication, those tables will be
ignored and omitted from the publication. They will be added to temporary tables and marked for replication during creation of the cutover publication.

**[IMPORTANT]** Don’t move on to next step until the **DataReplicationStatus** property for this step is shown as completed.

8. **Cutover: Remove non-primary key publication and temporary tables**

**[IMPORTANT]** This step only applies to customers using **AX 2012 Database Upgrade Toolkit for Dynamics 365** app version 8 or earlier.

This step performs the following actions:

a. Clean up the temporary tables that were created for non-primary key tables in the source database.

b. Delete the **AX_PUB_NoPKTable** publication.

You can validate the replicated data by using the ‘dv’ option. If there are mismatched tables, this step lets you create publications for them. If you want to exclude any mismatched tables for replication, close the app, and add those tables in **Data/IgnoreTables.xml**. Then rerun the app, and use the ‘dv’ option.

To learn more about the ‘dv’ option, see the **Reporting section of the application** section later in this topic.

9. **Cutover: Create constraint for non-primary key tables**

**[IMPORTANT]** This step only applies to customers using **AX 2012 Database Upgrade Toolkit for Dynamics 365** app version 8 or earlier.

This step extracts constraints for the non-primary key tables from the source database and creates them in the target database.

10. **Cutover: Remove replication setup**

This step deletes all the publications that were created in the source database, the distribution database, and the replication snapshot.

**NOTE**

To remove the **Snapshot** folder without causing an exception, run the following script in the source database.

Even if you don't run this script, you can ignore the exception message that you receive.

```sql
EXEC master.dbo.sp_configure 'show advanced options', 1
RECONFIGURE WITH OVERRIDE
EXEC master.dbo.sp_configure 'xp_cmdshell', 1
RECONFIGURE WITH OVERRIDE
```

11. **Post-replication: Update environment state to Replicated**

This step changes the state of the LCS environment from **Replication in progress** to **Replication completed**.

12. **Data upgrade: Trigger upgrade**
This step triggers the data upgrade. When the action is successful, the state of the LCS environment changes from **Replication completed** to **Data upgrade in progress**.

At this point, only the data upgrade trigger occurs. The actual data upgrade occurs in the self-service environment. To learn the status of the data upgrade, use the ‘ds’ option. To learn more about this option, see the Reporting section of the application section later in this topic.

If data upgrade is successful, the ‘ds’ option is shown as **AX 2012 upgrade topology (LCS) status: Deployed**, and all the upgrade steps will be in a **Completed** state.

If data upgrade fails, the ‘ds’ option is shown as **AX 2012 upgrade topology (LCS) status: Failed**, and one or more upgrade steps will be in a **Failed** state. The **Menu option (12)** tool will show a status of **Resume**.

After you address and fix the reasons for the failure, you can perform the **Resume** operation. When the action is successful, the state of the LCS environment will change from **Failed** to **Data upgrade in progress**.

NOTE
Repeat this step until the data upgrade is successful.

13. **Rollback data upgrade: Trigger rollback**

This step triggers the rollback of data upgrade. This rolls back the data to the point before the upgrade is triggered and sets the LCS environment state to **Replicated**. This will change the environment from **Failed** to the **Replicated** state.

At this point, you have only triggered the rollback. To see the rollback status, use the ‘rbs’ option. To learn more about this option, see the Reporting section of the application later in this topic.

If rollback is successful, the ‘rbs’ option is shown as **AX 2012 upgrade topology (LCS) status: Replicated**.

If rollback fails, the ‘rbs’ option is shown as **AX 2012 upgrade topology (LCS) status: Failed**.

For more information about the data upgrade process, see Upgrade from AX 2012 – Data upgrade FAQ. This topic answers some frequently asked questions about data upgrade during an upgrade from Microsoft Dynamics AX 2012.

**Reporting section of the application**

You can use the following options to review the reports of the replication validation, replication status, data upgrade status, and rollback data upgrade status.

- **dv) Report**: Validate the replication.

  This option compares the number of tables and records in the source server database and the target server database, and then shows the report. You should use this option only after step 8 is completed.

  If there are mismatched tables, this step lets you create a publication for them. If you want to exclude any mismatched tables for replication, close the app, and add those tables in **Data/IgnoreTables.xml**. Then rerun the app, and use the ‘dv’ option.

  You can find the report data at **output/PostValidationInfo.csv**.

- **rs) Report**: Get the replication status.
This option shows the report of the replication process for the publications that were created. You should use this option only after step 5 is started (that is, during the replication process for any publication).

- **ds) Report**: Get the data upgrade status.
  
  This option shows the report of the data upgrade process. You should use this option only after step 12 is started.

- **rbs) Report**: Get the rollback status.
  
  This option shows the report of the rollback process. You should use this option only after step 13 has started.

**Tooling section of the application**

- **Reset-rep**: Reset the replication setup by removing all the replication configurations. Publications and the distribution database are deleted. The status of all Replication and Cutover menu options is reset from Completed mode to Reset mode to help you redo the replication from the beginning.
- **Reset-all**: Reset all the menu options, and remove the replication configurations. The status of all the options is changed to Not Started.
- **Clear**: Clear the environment setup activity. All information is cleared from the cache, such as the project-Id value, Environment-Id value, and source database details. The status of step 1 is changed to Not Started.
- **Help**: Show the data upgrade migration options with the updated status.
- **Exit**: Close the application.

**Troubleshooting**

- **Scenario 1**: The migration app is prompting you to enter Project-Id and Environment-Id values.

  **Solution**: The user should be part of the project and should be assigned to one of the following roles: ProjectOwner, EnvironmentAdmin, or OperationsAdmin.

- **Scenario 2**: Migration app database connectivity failed for the source database server or the target database server.

  **Solution**: In the migration app, complete step 1, Data upgrade preparation: Environment setup activity.

- **Scenario 3**: The snapshot for any of the publications failed. This failure can be tracked in the Replication Monitor.

  **Solution**: In the Replication Monitor, on the Agents tab, select the failed publication, select and hold (or right-click) the snapshot agent, and then select Start agent to generate a snapshot.

- **Scenario 4**: If one of the steps fails in the migration app, and you must rerun that step, follow these steps:
  1. Close the migration app.
  2. In the migration app folder, find the Data folder.
  4. In the file, you can see all the menu options that have the same ID sequence. Find the step that you want to rerun, and update the Status value to 0.
IMPORTANT

Don't change anything else in this file. When you update the file, make sure that the migration app isn't in a running state.

5. Open the migration app, and run the step.

- **Scenario 5:** After the publication is created, the replication job fails, and the following exceptions occur:
  
  - **Exception 1:**
    
    Cannot execute as the database principal because the principal "dbo" does not exist, this type of principal cannot be impersonated, or you do not have permission. (Source: MSSQLServer, Error number: 15517)
    
    Get help: [http://help/15517](http://help/15517)
  
  - **Exception 2:**
    
    The process could not execute 'sp_replcmds' on 'replicationsrv\MSSQLSERVER2016'. (Source: MSSQL_REPL, Error number: MSSQL_REPL20011)
    
    Get help: [http://help/MSSQL_REPL20011](http://help/MSSQL_REPL20011)

    Cannot execute as the database principal because the principal "dbo" does not exist, this type of principal cannot be impersonated, or you do not have permission. (Source: MSSQLServer, Error number: 15517)
    
    Get help: [http://help/15517](http://help/15517)

  **Solution:** In SQL Server Management Studio (SSMS), open a query window, connect to the source database, and run the following command:

  ```sql
  EXEC sp_changedbowner 'sa'
  ```

- **Scenario 6:** The LCS status is **Failed**. However, in the migration app, the data upgrade trigger is successful.

  **Solution:** In the migration app, run the 'ds' option. This option reads the LCS environment state and the data upgrade status for every step and substep.

  **NOTE**

  If the data upgrade status and the LCS environment status are **Failed**, the status of step 12 in the Complete the data replication and upgrade procedure will be updated to **Resume**. The user can then resume the operation from the point where the upgrade process failed.

- **Scenario 7:** If you want to skip the failed step (if that step was manually run) and proceed with further steps, follow these steps:

  1. Close the migration app.
  2. In the migration app folder, find the **Data** folder.
  3. In the **Data** folder, open the **ReplicationMenu.Json** file.
  4. In the file, you can see all the menu options that have the same ID sequence. Find the step that you
want to rerun, and update the **Status** value to 1. By changing the status to 1, you mark the step as completed.

**IMPORTANT**

Don't change anything else in this file. When you update the file, make sure that the migration app isn't in a running state.

- **Scenario 8**: To migrate from an old version to the new version of the console app, follow these steps:

  1. Download the latest version of the console app from LCS.
  2. Take the `paramsdata.txt` and `ReplicationMenu.json` files from the old version of the console app, and put them under the same paths in the new version of the console app.
  3. Rerun the app.

- **Scenario 9**: The replication status for any of the publications is shown as **Waiting for snapshot to complete** for more than two hours.

  **Solution**: In the Replication Monitor, select and hold (or right-click) the publication, and then select **Reinitialize Subscription**.

- **Scenario 10**: You want to resume the data upgrade.

  **Solution**: The data upgrade status might not have been updated in the console app. Follow these steps to resume the data upgrade:

  1. To learn the status of the console app, perform the **Help** option. This option lists all the menu options and shows the current state.
  2. In the **Complete the data replication and upgrade** procedure, if the status of step 12 is **Successful**, run the 'ds' option in the migration app. This option updates the data upgrade status.

  After the 'ds' option is run, two types of status will be listed: the LCS environment status and the data upgrade status.

  - **Case 1**: If the LCS environment status is **Failed**, and the last step of the data upgrade is **Failed**, step 12 will show the Resume option.
  - **Case 2**: If the LCS environment status is **Failed**, and the last step of the data upgrade is **Completed**, step 12 will show the Resume option.
  - **Case 3**: If the LCS environment status is **Deployed**, and the last step of the data upgrade is **Completed**, step 12 will show **Successful**.
  - **Case 4**: If the LCS environment status is **Deployed**, and the last step of the data upgrade is **In Progress**, step 12 will show **Successful**, because the data upgrade job is running in the background.

- **Scenario 11**: After creating the publication, if the snapshot creation fails with the following error.
Learn about the replication configuration and status via SQL Server Management Studio

Solution: In the Replication Monitor, select and right-click the failed publication, and then select Generate Snapshot.

In SSMS, if Object Explorer includes a Replication folder, the replication feature is installed on the server and available.

After step 3 of the data upgrade process is completed, you should find the publisher configured under the Replication folder. To learn the replication status, select and hold (or right-click) the Replication folder, and then select Launch Replication Monitor.

- In the replication monitor, you can view all the publishers that have been created for replication.
- On the Snapshot tab, you can view the status of the snapshot.
- To view the detail log/transaction, double-tap (or double-click) a grid item.
- To view the data replication to the target, on the All Subscription tab, double-tap (or double-click) the subscription from the grid item.
This topic describes the process for upgrading Microsoft Dynamics AX 2012 databases to Dynamics 365 Finance + Operations (on-premises) version 10.0.x. Currently, upgrade is supported only from either Dynamics AX 2012 R2 or Dynamics AX 2012 R3.

### IMPORTANT

This topic explains the process for doing a data upgrade only. For information about how to do a code upgrade, see the upgrade guides that are available for cloud versions. The code upgrade tooling is available only through Microsoft Dynamics Lifecycle Services (LCS).

### AX 2012 upgrade to Dynamics 365 Finance + Operations (on-premises)

Two upgrade methods are currently supported:

- **Upgrade from inside the VHD** – This method involves copying your database into the virtual hard disk (VHD) and running the upgrade from inside the VHD. Overall, this method is easier.

- **Upgrade where the VHD points to your database** – This method involves pointing the VHD upgrade process to your database. The upgrade process is still run from inside the VHD.

### NOTE

The VHD doesn't require external network access to run the upgrade process.

### Prerequisites

1. Sign up for a preview subscription.

2. For each AX 2012 release, update to the most recent cumulative update that is available before you upgrade to the most recent Finance + Operations application release.

3. Install the pre-upgrade checklist. For more information, see [Installation](#).

4. Go through the data upgrade preparation steps. You can skip the “Set up user mapping” step. This step is relevant only for cloud-hosted upgrades.

5. Make a backup of your database (MicrosoftDynamicsAX). For more information, see [Create a Full Database Backup](#).

6. In LCS, go to the Shared asset library by selecting the tile on the right side of the page. Then, under **Select asset type**, select **Downloadable VHD**, and download all parts of the VHD package that most closely matches the version that you will upgrade to in your on-premises environment. The image requires a large amount of disk space. Therefore, be sure to download and extract the package on a drive that has enough free space.

7. The files that you downloaded are a self-extracting zip file. Extract the VHD to a location that has a good amount of free space.
8. Use Hyper-V to start a virtual machine (VM) and attach the VHD. (Note that the VM must be Generation 1.)

9. Connect to the VM. For information about the credentials, see Running the Virtual Machine (VM) locally.

10. Depending on your planned on-premises target version of 10.0.x and the VHD image that you downloaded, you might have to download and apply the required application update and platform update from the Shared asset library. Under Select asset type, select Software deployable package. For more information, see Install deployable packages from the command line.

   **IMPORTANT**
   In any case, make sure that you've applied the most recent quality update to your VHD, to ensure that it contains the most recent fixes for doing data upgrades.

11. If you have any extensions or customizations, install them on the VHD now. Otherwise, the upgrade process will remove any data that is related to customizations. If you must prepare your environment before the upgrade, check with your independent software vendor (ISV) or value-added reseller (VAR).

   **Upgrade from inside the VHD**
   1. Restore the backup that you created to the OneBox VM. For more information, see Restore a Database Backup Using SSMS.

   2. Optional: If the name of your restored database isn't AXDB, open Windows PowerShell as an administrator, and run the following script.

   ```powershell
   .\Configure-On-Premises-Upgrade.ps1 -DatabaseName '<DB-name>'
   ```

   **NOTE**
   Replace `<DB-name>` with the name of your database (for example, AXDB). If you want to edit more values, see the appendix later in this topic.

   The script will run a database connection test to verify that the information that you provided is valid.

   3. In LCS, go to the Shared asset library. Under Select asset type, select Software deployable package, and then select AX2012DataUpgrade-10-0-8 to download the MajorVersionDataUpgrade.zip file.

   4. Copy the file, paste it in the desired location (for example: c:\D365FFOUpgrade\), and unzip it.

   5. Open a Command Prompt window as an administrator, change the directory to the folder that you just unzipped, and run the following commands.

   ```powershell
   AxUpdateInstaller.exe generate -runbookid=upgrade -runbookfile=upgrade.xml -topologyfile=defaulttopologydata.xml -servicemodelfile=defaultservicemodeldata.xml
   AxUpdateInstaller.exe import -runbookfile=upgrade.xml
   AxUpdateInstaller.exe execute -runbookid=upgrade
   ```

   6. After the upgrade process is successfully completed, back up the newly upgraded database. If you have customizations from ISVs or VARs, check whether you must run some post-data upgrade scripts.

   7. Restore the database into your on-premises environment's SQL Server, but give it a name that differs from the name of the AX 2012 database (for example, name it AXDBUpgraded). The restored database must be configured. Follow the steps in Configure the Finance + Operations database.

If you have customizations, follow these steps:

a. In LCS, go to the Shared asset library.

b. Under Select asset type, select Model, and then download Dynamics 365 Finance + Operations on-premises, Version 10.0.x Demo Data. Select the version that is closest to the 10.0.x environment that you will deploy as the on-premises baseline.

c. Use the restore backup option for SQL Server to create a new database from this file. (Typically, this database is named AXDB.) For more information, see Restore a Database Backup Using SSMS.

d. The demo database must be configured. Follow the steps in Configure the Finance + Operations database.

e. In LCS, set up a new environment, and deploy it with version 10.0.x. For more information, see Set up and deploy on-premises environments (Platform update 12 and later). When you deploy the environment, the name of the database that you specify should be the name of the database that you created earlier (typically AXDB).

f. Apply your own customizations, and ISV and VAR modules, to the newly created 10.0.x environment. Otherwise, when the environment is initially synced with the database, it will delete any customization-related or extension-related data.

If you don't have customizations, follow these steps:

a. Optional: Rename your old database (typically AXDBold), and then rename your new database (typically AXDB). In the next step, make sure that you enter the name of the upgraded database.

b. In LCS, set up a new environment, and deploy it with version 10.0.x (Redeploy). For more information, see Set up and deploy on-premises environments (Platform update 12 and later).

**Upgrade where the VHD points to your database**

1. Back up the database from your on-premises environment (typically AXDB). For more information, see Create a Full Database Backup (SQL Server).

2. Restore the backup that you just created into the database server, and give it a different name (for example, AXDBtoupgrade). For more information, see Restore a Database Backup Using SSMS.

3. Open Windows PowerShell as an administrator, and run the following script.
.\Configure-On-Premises-Upgrade.ps1 -DatabaseName 'DB-name' -DatabaseServer 'SqlServerName' -DatabaseUser 'User' -DatabasePassword 'Password'

NOTE
- Replace <DB-name>, <SqlServerName>, <User>, and <Password> with the values that you require.
- Only SQL Server authentication is officially supported for this upgrade. For more information, see Create a Database User.
- You must add the certificate authority certificate that signed your SQL Server certificate to the trusted certificate authorities store in your Onebox VHD. For more information, see Installing the trusted root certificate.
- Make sure that the database user that you use has the sysadmin server role, or at least All Privileges, assigned on the database that you want to upgrade. Also make sure that the user has permissions to access tempDB. Step 6 of the upgrade process will fail if these conditions aren't met.
- When you install the certificate authority certificate in the OneBox VHD, make sure that you use the fully qualified domain name (FQDN) or IP address to connect to the database that appears there. If you can't access the database by using the domain name, because it doesn't point to that server, edit your hosts file, and add the FQDN and the IP address that the FQDN should be resolved to.

4. In LCS, go to the Shared asset library. Under Select asset type, select Software deployable package, and then select AX2012DataUpgrade-10-0-8 to download the MajorVersionDataUpgrade.zip file.

5. Copy the file, paste it in the desired location (for example: c:\D365FFOU...4\), and unzip it.

6. Open a Command Prompt window as an administrator, change the directory to the folder that you just unzipped, and run the following commands.

   a. AxUpdateInstaller.exe generate -runbookid=upgrade -runbookfile=upgrade.xml -topologyfile=defaulttopologydata.xml -servicemodelfile=defaultservicemodeldata.xml
   b. AxUpdateInstaller.exe import -runbookfile=upgrade.xml
   c. AxUpdateInstaller.exe execute -runbookid=upgrade

7. If you have customizations from ISVs or VARs, check whether you must run some post–data upgrade scripts.

8. Run the Configure-OnpremUpgrade.ps1 script by using the values that are stated in the Resetting the VHD database (Optional) section later in this topic.

9. Configure your upgraded database for Finance + Operations by following the steps in Configure the Finance + Operations database.

10. Deploy a new Dynamics 365 Finance + Operations (on-premises) environment.
    - If you have customizations, follow these steps:
      a. In LCS, go to the Shared asset library.
      b. Under Select asset type, select Model, and then download Dynamics 365 Finance + Operations on-premises, Version 10.0.x Demo Data. Select the version that is closest to the 10.0.x environment that you will deploy as the on-premises baseline.
      c. Use the restore backup option for SQL Server to create a new database (typically AXDB) from this file. For more information, see Restore a Database Backup Using SSMS.
      d. The demo database must be configured. Follow the steps in Configure the Finance + Operations database.
      e. In LCS, set up a new environment, and deploy it with version 10.0.x (Redeploy). For more
Configuring existing users

If you followed either of the previous procedures, you can sign in by using the Administrator user that you specified in LCS. However, none of your other users can sign in until they have been configured for the new system. Run a `Select` statement against your USERINFO table, and make a note of the value in the `NETWORKDOMAIN` field for the Administrator user (for example, `https://adfs.contoso.com/adfs` or `http://adfs.contoso.com/adfs/services/trust`). Then, for all interactive users who should be able to sign in, set the `NETWORKDOMAIN` field to the same value that the Administrator user has. The `NETWORKALIAS` field must also be modified. In Finance + Operations, this field is set to the user’s email address (for example, `testuser@contoso.com`).

Reseting the VHD database (Optional)

If you used the `Configure-On-Premises-Upgrade.ps1` script, run the following command to reset your database to the default configuration.

```
\Configure-OnPremUpgrade.ps1 -DatabaseName 'AXDB' -DatabaseServer 'localhost' -DatabaseUser 'axdbadmin' -DatabasePassword 'AOSWebSite@123'
```

Appendix

Using the Configure-On-Premises-Upgrade.ps1 script
This script is intended to be run only from a OneBox VHD environment. The script requires that you pass at least the `DatabaseName` parameter. If you don’t pass this parameter, the script automatically requests it.

You can pass an additional parameter, such as `DatabaseServer` or `DatabaseUser`, if you want. However, in this case, the script will request all additional parameters. This behavior occurs because the script will assume that you want to point the database connection to a machine outside the VM. Therefore, those parameters are required to correctly establish the connection.

The following parameters can be passed to the script:

- `-DatabaseName` – The name of the database to upgrade.
- `-DatabaseServer` – The database server that contains the Finance + Operations database.
- `-DatabaseUser` – The user name for SQL Server Authentication.
- `-DatabasePassword` – The password for SQL Server Authentication.

After the configuration has been passed, the script uses the new parameters to run a database connection test. If the script can’t connect to the database, we recommend that you debug the connection from SQL Server Management Studio or another tool.

```powershell
<#
.Synopsis
    Configures a OneBox deployment to upgrade an OnPrem 7.x database to OnPrem 10.0.x

.DESCRIPTION
    This must be executed before the upgrade process is carried out.

.EXAMPLE
    .\Configure-OnPremUpgrade.ps1 -DatabaseName 'AxDB'

    .\Configure-OnPremUpgrade.ps1 -DatabaseName 'AxDB' -DatabaseServer '127.0.0.1' -DatabaseUser 'axdbadmin'
    -DatabasePassword 'secretPass'
#>
[CmdletBinding()]
param
(
    # Database server containing Microsoft Dynamics 365 for Operations, on-premises database.
    [AllowNull()]
    [string] $DatabaseServer,

    # Database name that you want to upgrade.
    [Parameter(Mandatory = $true)]
    [string] $DatabaseName,

    # Username for SQL Authentication.
    [AllowNull()]
    [string] $DatabaseUser,

    # Password for SQL Authentication.
    [AllowNull()]
    [string] $DatabasePassword
)
$webroot = "C:\AOSService\webroot"
$commandParameter = " -decrypt "$webroot\web.config""
```
$command = Resolve-Path "$webroot\bin\Microsoft.Dynamics.AX.Framework.ConfigEncryptor.exe"

Start-Process $command $commandParameter -PassThru -Wait

if([string]::IsNullOrEmpty($DatabaseUser) -and [string]::IsNullOrEmpty($DatabasePassword) -and [string]::IsNullOrEmpty($DatabaseServer)) {
    [xml]$web = Get-Content $webroot\web.config

    $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.Database']").value = [string]$DatabaseName
}
else {
    if([string]::IsNullOrEmpty($DatabaseServer)) {
        $DatabaseServer = if($value = Read-Host 'What is the IP or FQDN of the Database server? [127.0.0.1]') {$value} else {'127.0.0.1'}
    }

    if([string]::IsNullOrEmpty($DatabaseUser)) {
        $DatabaseUser = if($value = Read-Host 'What is the SQL Authentication username? [axdbadmin]') {$value} else {'axdbadmin'}
    }

    if([string]::IsNullOrEmpty($DatabasePassword)) {
        $dbPassEn = if($value = Read-Host 'What is the SQL Authentication password?' -AsSecureString) {$value} else {''}
        $BSTR = [System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($dbPassEn)
    }

    [xml]$web = Get-Content $webroot\web.config

    $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.Database']").value = [string]$DatabaseName

    #Save Configuration to webroot config
    $web.Save("$webroot\web.config")
}

#Reloading the configuration to run test
[xml]$web = Get-Content $webroot\web.config

$TestDbServer = $web.SelectCommandSingleNode("configuration/appSettings/add[@key='DataAccess.DbServer']").value
$TestDbName = $web.SelectCommandSingleNode("configuration/appSettings/add[@key='DataAccess.Database']").value
$TestDbUser = $web.SelectCommandSingleNode("configuration/appSettings/add[@key='DataAccess.SqlUser']").value
$TestDbPass = $web.SelectCommandSingleNode("configuration/appSettings/add[@key='DataAccess.SqlPwd']").value
# Setting up connection test.

$dbConn = New-Object System.Data.SqlClient.SqlConnection
$dbConn.ConnectionString = "Data Source=$TestDbServer;User ID=$TestDbUser;Password=`"$TestDbPass`";Database=$TestDbName"

try{
    $dbConn.Open()
    $result = $true
}
catch{
    $result = $_.Exception.Message
}
Finally{
    $dbConn.Close()
}

$commandParameter = " -encrypt `"$webroot\web.config`""
Start-Process $command $commandParameter -PassThru -Wait

if($result -ne $true){
    Write-Host "nThe connection to the Database Server failed:" -ForegroundColor Red
    Write-Host $result -ForegroundColor Red
}
else{
    Write-Host "nThe connection to the Database Server was successful!" -ForegroundColor Green
}

## Troubleshooting

- Exception calling "Open" with "0" argument(s): "Cannot open database "AxDB1" requested by the login. The login failed. Login failed for user 'axdbadmin'." You supplied the Wrong database name or the user doesn't have access to that database.
- Exception calling "Open" with "0" argument(s): "A network-related or instance-specific error occurred while establishing a connection to SQL Server. The server was not found or was not accessible. Verify that the instance name is correct and that SQL Server is configured to allow remote connections. (provider: Named Pipes Provider, error: 40 - Could not open a connection to SQL Server)". The script could not establish a connection with the SQL Server specified. Check the ip/fqdn and port that you used.
- Exception calling "Open" with "0" argument(s): "Login failed for user 'axdbadmin'." The supplied login credentials are not correct.
Cutover is the term that we use for the final process of getting a new system live. The cutover process consists of the tasks that occur after Microsoft Dynamics AX 2012 is turned off, but before Finance and Operations is turned on. The purpose of upgrade cutover testing (mock cutover) is to practice the cutover process, to help guarantee a smooth experience for everyone who is involved during the actual cutover to go-live.

There are three main workstreams during a cutover:

- **Technical workstream** – This workstream includes the data upgrade execution process. Your business will enforce a limit on the amount of downtime that is allowed. During this downtime, neither AX 2012 nor Finance and Operations will be available. This workstream might have to tune the data upgrade procedure to meet the business’s downtime limit.

- **Functional workstream** – This workstream includes the configuration tasks that are performed after the data upgrade is completed. All these tasks must be documented and quantified, and a resource must be assigned, because both the functional workstream and the technical workstream must fit within the business’s downtime limit.

- **AX 2012 rollback** - This workstream includes rolling back to an AX 2012 environment. Although it’s unlikely that you will have to roll back, it’s very important that you have a tested process in case you require it.

The following illustration shows the overall process for cutover to go-live as it will occur in the production environment.

The mock cutover process is very similar to data upgrade validation in a sandbox environment. We assume that you are familiar with that process, and have already performed it. Mock cutover differs in the following ways:

- After you perform a data upgrade in the sandbox environment, see the instructions in **Self-service database refresh** to copy your upgraded database from the data upgrade sandbox environment into your production environment.

- We added the following tasks:
Technical workstream

The technical workstream involves various technical team members: the database administrator (DBA), the AX 2012 system administrator, server administrators, and developers who are familiar with AX 2012 and Finance and Operations.

During cutover testing, the technical team is focused on performance and reliability testing of the data upgrade process, to make sure that it meets the business’s downtime limit. Many elements of hardware and software are involved in this process. Some of these elements are on-premises, whereas others are in the Microsoft cloud. In addition, many elements of custom application code and standard code are involved. The result of this testing should be confidence in the cutover process for your environment.

Technical workstream process

For the technical workstream, the cutover testing process is the same described in Upgrade from AX 2012 - Data upgrade in self-service environments.

Functional workstream

After data upgrade, several configuration tasks will be required in the new environment. The goal of this workstream is to document and quantify all configuration tasks, and to assign a resource to each task, to help guarantee that these tasks can be done together with the technical workstream during the downtime window.

Typically, functional tasks involve changing the values of specific system parameters or other configuration data. These tasks are identified through the full functional test pass, which is a separate activity from the cutover testing. When a task of this type is identified, it should be reviewed together with the functional resource and your developer.

Larger changes might require that a new custom data upgrade script be written to update the data during the data upgrade process. However, the functional resource can manually run smaller changes through the new system after data upgrade.

Larger changes that have new data upgrade scripts must be tested. Therefore, one or more additional iterations of the MajorVersionDataUpgrade.zip package will have to be run. It’s important that you weigh the cost of running the package again against the cost of manual data entry.

For each manual change, a task must be added to the cutover plan document. This task must show the following details:

- Perform a smoke test.
- Complete application setup tasks. This step can be large, depending on the functionality that is used. During this step, the functional team configures new application functionality so that it’s ready to be used in the upgraded system.
- Allow users back in. Notify your user base that the upgrade is completed and that they can use the system again.
• What is the task, and what must be done?
• Who must do it?
• How long does it take?

**Add users, and perform functional tests**
When you have fully configured your environment, add users, and perform appropriate testing.

**Roll back to AX 2012**
The goal of this task is to restore the database by using the backup that was made when AX 2012 was turned off, and then turn AX 2012 back on. The state of integrated systems might also have to be restored. However, because integrated systems vary from business to business, you must plan for this scenario independently, based on your specific circumstances. Although it's unlikely that you will have to roll back, it's very important that you have a tested process in case you require it.
This topic describes the tasks that you might have to perform in Finance and Operations apps, like Dynamics 365 Finance and Dynamics 365 Supply Chain Management, after you complete a code and data upgrade from Microsoft Dynamics AX 2012. A process data package (PDP) that is available in Microsoft Dynamics Lifecycle Services (LCS) includes links to the following menu items. This PDP will fill in the Data validation checklist workspace. The Data validation checklist workspace lets users track a project and monitor the tasks that are required in order to complete it.

Document management

If you use document management, existing documents or attachments that are stored in the database should be migrated to Microsoft Azure Blob storage. To complete this migration, use the Migrate files button on the Migrate files tab on the Document management parameters page. This operation is not critical as document management can still access file stored in the database, but the files can take considerable database storage and the retrieval is less efficient. The file migration process will migrate all possible database files to Microsoft Azure Blob storage, reporting on any failures and continuing. If any errors are reported, attempt running the file migration process again.

If the file migration process isn’t able to complete without failure, this may be that the files stored in the database are corrupt, which Microsoft is unable to repair. If this is the case, you can request a non-business critical support case be opened to enable conversion of the attachments into note records, which will retain any previous notes as well as the names of the files that were stored in the database. Note that the files themselves cannot be recovered.

Print management

If you use Print management, the references to network printers from AX 2012 won’t be valid. You must set up and reference network printers on the Document routing page. For more information, see Install the Document Routing Agent to enable network printing.

Commerce

After you complete the upgrade from AX 2012, you must configure registers and devices.

To configure a register, click Retail and Commerce > Channels > Stores. Select the row for the channel, and then expand the Registers FactBox. Click More, click New, and complete the setup of the register.

To configure a device, click Retail and Commerce > Channel setup > POS Setup, and then click New.

Additionally, you must run all jobs (9999) for the channel database. Click Retail and Commerce > Headquarters setup > Commerce scheduler > Channel database. Select the row for the appropriate channel database, and then click Full data sync. Select the 9999 (All jobs) distribution schedule, and then click OK. Click OK again to run the job.
Service industries

After you complete the upgrade from AX 2012, you must set up resource capacity roll-up and project ledger intercompany posting.

To run the Resource capacity roll-up batch job, click Project management and accounting > Inquiries and reports > Capacity synchronization. You must run this batch job to set up the resource and resource calendar reservation data. This data will be required if you use project resource scheduling. For more information, see Project resourcing.

To enable project ledger intercompany posting, click Project management and accounting > Setup > Posting > Ledger posting setup. On the Cost accounts tab, in the Ledger account types field, select Intercompany cost, and then enter the details of the lending legal entity. On the Revenue accounts tab, in the Ledger account types field, select Intercompany revenue, and then enter details of the borrowing legal entity.

Budget planning

After you complete the upgrade from AX 2012, you must set up Budget planning columns and layouts. To complete this setup, click Budgeting > Setup > Budget planning > Budget planning configuration.

Additionally, you must update Budget planning processes so that they use the appropriate layout for each budget stage. To update Budget planning processes, click Budgeting > Setup > Budget planning > Budget planning process.

For more information about Budget planning upgrade, see Upgrade budget planning.
IMPORTANT

Upgrade is currently only supported from either Dynamics AX 2012 R2 or Dynamics AX 2012 R3. For each release, please update to the latest available cumulative update before upgrading to latest Finance and Operations application release.

After data upgrade is completed, we recommend that you complete a full functional test pass of all business processes. In a full functional test pass, you do an extensive retest of all business processes that are performed by using Finance and Operations. Tests should include both processes that have been brought forward from Microsoft Dynamics AX 2012 and new processes that use features from Finance and Operations.

Depending on your code quality, bug remediation and retesting might require several iterations of the functional test pass. After a bug is fixed, take care to retest all processes that are involved, to make sure that no downstream or upstream processes are affected by the change.

Old data vs. new data

As you create a list of tests to perform, consider the impact of old data versus new data. This type of testing will help you uncover data-related bugs. The code that created the old and new data might be very different, and this difference can be a common root cause of bugs.

For example, if the business process that you’re testing is cancellation of a purchase order, can you cancel a purchase order that was started before it was upgraded to the new system? Can you also cancel a purchase order that was started after the upgrade to the new system?

We used a very simple example here, but the testing requirement can be more complex, because many business processes in the system are interconnected, and the effect of old data versus new data effect is cumulative.

For example, here are the stages of a production test flow:

1. The item master is designed and released to a legal entity.
2. Item requirements are created.
3. Production orders are generated.
4. Purchase orders are generated.
5. Production order processing occurs (shop floor).
6. Vendor payment (payment of purchase orders) occurs, and so on.

In this production test flow, each stage can be performed by using either new records or old records as input. The result is a matrix of tests that covers every combination of old and new data. For some processes, test matrices might seem excessive, and they might actually be excessive in practice. Therefore, you can decide to focus on certain combinations that you predict will be used the most. However, it’s still helpful for you to know what you aren’t covering. Make a conscious decision, where you know what you have, what you’re going to focus most of the testing on, and what you’re not going to focus on.
IMPORTANT

Upgrade is currently only supported from either Dynamics AX 2012 R2 or Dynamics AX 2012 R3. For each release, please update to the latest available cumulative update before upgrading to latest Finance and Operations application release.

As the go-live date approaches, it’s important that you implement this series of steps to help ensure that the source Microsoft Dynamics AX 2012 system and the upgrade process both remain stable and consistent for go-live.

As part of this process, you must lock down any further code changes or application setup changes before you run the final cutover test.

Code freeze

All code changes in the AX 2012 environment should be frozen. Implement an escalation process to handle any critical issues that appear in AX 2012. By default, any new code changes that are required should be implemented only in the new system, not in the AX 2012 environment. Implementation of proposed code changes in the AX 2012 environment should be discussed at the management level. If a code change is made in AX 2012, the same change must be made in the new system. In that case, another iteration of cutover testing and functional testing might be required.

Application configuration freeze

All application configuration changes should be frozen in the AX 2012 environment, because these changes could affect how the new system behaves or how the data upgrade scripts behave. Configuration changes are changes in the AX 2012 application that are related to the configuration of system functionality. By freezing these changes, you help guarantee the stability of the data upgrade process.

Because configuration changes are typically controlled by the AX 2012 system administrator or a small group of trusted super users, we don’t recommend that you enforce the freeze by changing security access. Instead, implement the freeze through a business process that is communicated to those users. Changes to security access might require a code change (changes to the role definitions themselves) or a configuration change (reassignment of users), and these changes could affect the upgrade process.

Running the final cutover test

After no further code or setup changes will occur, run a final cutover test to make sure that all data and code upgrade tasks still run as expected.

NOTE

You must complete this step even if you freeze code and setup changes at the beginning of the upgrade project, because the data itself changes every day. This final cutover test also validates that the current data is upgraded successfully.

Make sure that functional testing is performed against this last upgraded copy.

At this point in the upgrade project, we recommend that you categorize any bugs that are found:
- **Blocking** – The upgrade project can’t proceed until every bug of this type is fixed. The upgrade must be postponed, and can proceed only after the bug is remediated and the cutover test is run again. For a bug to be classified as blocking, it must meet these conditions:
  - It prevents a critical business process from being completed.
  - No workaround or mitigation is available for it.
- **Non-blocking** – The upgrade project can proceed. Bugs of this type can be fixed in the upgraded system.
IMPORTANT
Upgrade is currently only supported from either Dynamics AX 2012 R2 or Dynamics AX 2012 R3. For each release, please update to the latest available cumulative update before upgrading to latest Finance and Operations application release.

After you have successfully completed upgrade testing in a Standard or Premier Acceptance Test environment (Sandbox Tier 2 or higher), and you have also completed a successful test cutover, the time has arrived to upgrade your production environment and go live.

Cutover is the term that we use for the final process of getting a new system live. This cutover process consists of the tasks that occur after Microsoft Dynamics AX 2012 is turned off but before Finance and Operations is turned on. Before you plan your final cutover, you need to successfully complete one successful mock cutover as described in Cutover testing.

The following illustration shows the overall process for cutover to go-live as it will occur in the production environment.

NOTE
In this topic, we use the term sandbox to refer to a Standard or Premier Acceptance Testing (Tier 2 or 3) or higher environment connected to a SQL Azure database.

Overall process
The high-level steps of the production environment upgrade process are the same as the Mock cutover process, refer to Upgrade from AX 2012 - Cutover testing (Mock cutover) for detailed instructions.

1. Be sure you've completed the pre-upgrade checklist for data upgrade and custom code is deployed in a sandbox environment. The sandbox environment must only be used for data upgrade.

2. Download the AX 2012 Database Upgrade Toolkit for Dynamics 365 from Microsoft Dynamics Lifecycle Services (LCS) in the Shared asset library > Model area. Use this toolkit from the source SQL Server.

3. Execute replication setup and keep monitoring it on a regular basis using the toolkit.

4. Turn off the AX 2012 AOS instances at the time of downtime/cut-over.

5. Ensure replication completes. Validate replication completion by comparing the number of records between...
source and target using this toolkit. For more information about how to validate the replication, see Reporting section of the application.

6. Execute cutover steps using the toolkit and ensure its completion.
7. Trigger the data upgrade using the toolkit and finish the data upgrade.
8. Use Self-service database refresh process to copy your upgraded database from the sandbox environment into your production environment.
9. Complete application configuration and complete smoke test.
10. Allow users to access the Finance and Operations app again.

Prerequisites

Before you can perform an upgrade in the production environment, the following prerequisites must be met:

- Complete the code upgrade and data upgrade in a sandbox environment and successfully complete a functional test pass.
- Deploy the production environment. Before the option to request the deployment of the production environment is an option, you must have completed:
  - The Subscription estimator in LCS. We use this to help us size your production environment because it provides details of the throughput you’ll require.
  - The Test phase of the methodology in LCS. This is to help ensure that you’re at the stage in your project where you’re ready to start testing in the production environment.
  - After a request is submitted to Microsoft to deploy the production environment, it will take roughly 24 hours to deploy, so ensure that you leave enough time for this to happen.
- Apply all necessary updates and customizations (AOT deployable packages) to the production environment. There should not be any code change after signing off on a Mock cutover.

Additional resources

- Onboarding
- Self-service database refresh
- Upgrade from AX 2012 - Cutover testing (Mock cutover)
This topic answers some frequently asked questions about data upgrade during an upgrade from Microsoft Dynamics AX 2012.

**Is the Tier 2 Azure SQL database sized enough for upgrades of large databases?**

If the database size grows, a Tier 2 sandbox that is deployed on an "elastic pool" should automatically be resized as required.

**Does the AX 2012 Database Upgrade Toolkit for Dynamics 365 support the Microsoft Government Community Cloud?**

No, the AX 2012 Database Upgrade Toolkit for Dynamics 365 doesn't currently support the Government Community Cloud (GCC).

**What type of validation is done as part of the AX 2012 Database Upgrade Toolkit for Dynamics 365?**

Few validations are done as part of the AX 2012 Database Upgrade Toolkit for Dynamics 365. For example, the toolkit validates that you've installed the required KBs (prerequisites) in AX 2012. If you haven't installed them, you can't start the replication process. There is also an option to run a record count check on the replicated data.

**What is the recommended approach if the source AX 2012 database is in a different region than the target database?**

For optimal replication performance, we recommend that the source AX 2012 database and the target database be in the same region. Customers can deploy the sandbox environment in the same region as the source and do the data upgrade. Then, after the upgrade is completed, the sandbox environment can be moved to the required region.

**Are the SQL BACPAC and DACPAC processes still supported for AX 2012 data upgrades in sandbox environments?**

No, the SQL BACPAC and DACPAC process are no longer supported for AX 2012 data upgrades in sandbox environments. Customers must use the AX 2012 Database Upgrade Toolkit for Dynamics 365 to do data upgrades in sandbox environments.

I've upgraded an AX 2012 database in a cloud hosted environment (dev) and uploaded the upgraded BACPAC file into Lifecycle Services. However, I receive an error message when I then try to import the BACPAC file into a sandbox environment. How do I fix the error?

When you try to import an upgraded BACPAC file from Microsoft Dynamics Lifecycle Services (LCS) into a sandbox environment, you might receive the following error message:
Importing AX 2012 bacpac file into Dynamics 365 environment isn't supported as it would result in a loss of the imported data and would put the environment in a failed state.

Validation is done to prevent a BACPAC file from being imported into a sandbox environment. For the AX 2012 database upload into a sandbox environment, you must use the AX 2012 Database Upgrade Toolkit for Dynamics 365, which utilizes SQL replication to transfer the data.

Can I filter on the table data that will be replicated (for example, to limit specific records only for replication to the target database)?

No, the AX 2012 Database Upgrade Toolkit for Dynamics 365 doesn't support filtering on the table data that will be replicated.
In Microsoft Dynamics AX 2012, you could use the same delimiter for your chart of accounts and dimension values. In current versions of Finance and Operations, you cannot have the same delimiter for the chart of accounts and dimension values. If there is a duplicate delimiter, you can change it after upgrade.

**Update delimiter**

If there is a conflict with the chart of accounts, the chart of accounts delimiter and the project/subproject ID format can be changed. No other dimension delimiters can be changed.

- You can change the chart of accounts delimiter after upgrade in **General ledger parameters > Chart of accounts and dimensions > Change delimiter**.
- If the only conflict is with the project/subproject ID format, you can change that value in **Project management and accounting parameters > General > Modify subproject format**.

**How to determine if your environment requires updated delimiters**

If delimiters in your upgraded environment are conflicting, you may experience instability when entering values in a segmented entry control or dimension entry control. This means that you will need to always use lookups or a flyout menu when entering account and dimension combinations.
This topic provides information about the Project resource scheduling data model.

Physical data model for Project resource scheduling

The following diagram represents the data design structure of the Project resource scheduling physical data model.

Tables

The following table provides a list of additional tables that support the Resource management data model.

<table>
<thead>
<tr>
<th>TABLE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResResourceIdentifier</td>
<td>Stores all resources and a subset of records from the WrkCtrTable table that are identified as resources. When a resource is added, a record will be added to the WrkCtrTable table. A record will also be added to this table with a Foreign Key reference to the new record in the WrkCtrTable table.</td>
</tr>
<tr>
<td>ResourceSetup</td>
<td>Specifies a resource’s property. This table replaced the ProjWorkerSetup table in Dynamics AX 2012.</td>
</tr>
<tr>
<td>ResBooking</td>
<td>Stores all resource booking type reservations. This table replaced some of the information stored in the PSASchedEmplReservation table in Dynamics AX 2012.</td>
</tr>
</tbody>
</table>
TABLE | DESCRIPTION
--- | ---
ResAssignment | Stores all resource assignment type reservations. This table replaced some of the information stored in the PSASchedEmpRes table in Dynamics AX 2012.
ResourceCategorySetup | Stores the resource default project role association that is validated by a time range. This is a time effective table which has a Foreign Key to ResResourceIdentifier.RecId and a Foreign Key to PSASchedRole.RecId with ValidFrom and ValidTo fields. The PSASchedRole table is used to store the project role.
ResCalendarCapacity | This is a data denormalized table based on WorkCalendarTable, WorkCalendarDate, and WorkCalendarDateLine to calculate resource capacity. The calculation is based on the resource associated calendar date line definition, which is specified in the WorkCalendarDataLine table. The data in this table can be generated by running the Synchronize resource capacity roll-ups batch job.
ResRollUpCalendar | This is a data denormalized table based on ResCalendarCapacity. The data in this table is used by the ResRollUpWriter class to speed up the process of adding resource records to the ResRollup table. The data in this table can be generated by running the Synchronize resource capacity roll-ups batch job.
ResRollUp | This is a data denormalized table based on ResBooking, ResAssignment, and ResRollUpCalendar for performance improvement. This table contains all of the capacity and reservation data for time scales, including Day, Week, Month, Quarter, and Half year, for all resources. This data is accessed by the AvailabilityView control, which is included with any resource scheduling X++ form that includes this control, like the Resource availability form. The data in this table can be generated by running the Synchronize resource capacity roll-ups batch job. Data is updated for every resource reservation action such as adding a new resource, booking, assignment, and reservation cancellation.

Views
The following table provides a list of the most informative views that you can use to access the Resource management data model.

VIEW | DESCRIPTION
--- | ---
ResResourceWorkCenterView | This view is based on ResResourceIdentifier joining to WrkCtrTable, which captures only the fields that are required for resource management.
ResourceView | This view is based on ResResourceIdentifier, which is only based on ResResourceWorkCenterView. In a future release it can be extended to more than ResResourceWorkCenterView if a resource record will be saved on more than just ResResourceWorkCenterView.
<table>
<thead>
<tr>
<th>VIEW</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResourceWorkerView</td>
<td>This view replaces the HcmWorker table, which is based on ResResourceIdentifier. WrkCtrTable has a worker field which is a Foreign Key to HcmWorkerTable. This view includes all WrkCtrTable resource which worker field is not 0.</td>
</tr>
<tr>
<td>ResourceLegalEntityView</td>
<td>This view replaces the HcmEmployment table. This is the resource legal entity (LE) view, which is a union of ResourceWorkerLegalEntityView and ResourceOtherLegalEntityView. The purpose of this view is to look up the resource legal entity as well as the associated valid from and valid to date. For example, an HcmWorker resource can be hired by multiple LEs with different ValidFrom and ValidTo dates. If the worker is hired by multiple LEs, then this worker resource will have multiple records in this view. If the resource is not a HcmWorker, then this resource will have only one record and validFrom = minDate and ValidTo = maxDate.</td>
</tr>
<tr>
<td>ResourceCalendarView</td>
<td>This view replaces WorkCalendarTable. This view contains the calendar RecId of the resource. This is a union view of ResourceWorkerCalendarView, ResourceOtherCalendarView, and ResourceGroupResourceCalendarView. To look up the resource calendar, users should first look up the ResourceLegalEntityView. This is because a resource should have a different calendar depending on the resource's legal entity. First, locate the record from ResourceLegalEntityView and get the RefRecId and RefTableId fields. Then, look up ResourceCalendarView by setting ResourceLegalEntityRefRecId = ResourceLegalView.RefRecId and ResourceLegalEntityRefTableId = ResourceLegalView.RefTableId.</td>
</tr>
<tr>
<td>ResourceCategoryView</td>
<td>This view shows the resource category (the resource role for a worker resource), which is based on the PSASchedRole table.</td>
</tr>
<tr>
<td>ResSkillHcmView</td>
<td>This view is based on HcmSkill to capture all the skills that are defined in the HcmSkills table.</td>
</tr>
<tr>
<td>ResCertificateHcmView</td>
<td>This view is based on HcmCertificateType to capture all the certificates that are defined on the HcmCertificateType table.</td>
</tr>
<tr>
<td>ResEducationHcmView</td>
<td>This view is based on HcmEducationDiscipline to capture all the education disciplines that are defined in the HcmEducationDiscipline table.</td>
</tr>
<tr>
<td>ResProjectHcmView</td>
<td>This view is based on ProjTable to capture all projects that exist in the ProjTable.</td>
</tr>
<tr>
<td>ResRoleView</td>
<td>This view is based on ResourceCategoryView to capture all resource categories.</td>
</tr>
</tbody>
</table>
### VIEW DESCRIPTION

<table>
<thead>
<tr>
<th>VIEW</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResCharacteristicView</td>
<td>This view is the union view of ResSkillHcmView, ResCertificateHcmView, ResEducationHcmView, ResProjectHcmView, and ResRoleView. This view is used to gather all the characteristics that are defined as selectable criteria for the resource search feature.</td>
</tr>
<tr>
<td>ResResourceCharacteristicView</td>
<td>This view is the union view of ResResourceCharacteristicHcmView, ResResourceResourceCategoryView, ResResourceNameView, and ResResourceRoleCharacteristicView. This view is used to identify the characteristics that are associated with a resource for the resource search feature.</td>
</tr>
<tr>
<td>ResCalendarCapacityView</td>
<td>This view is based on ResCalendarCapacity, which contains calendars that are specified in WorkCalendarTable with capacity details.</td>
</tr>
<tr>
<td>ResCapacityView</td>
<td>This view shows the resource's capacity per hour.</td>
</tr>
<tr>
<td>ResResourceCapacityWorkDaysView</td>
<td>This view is based on ResCapacityView, which shows the resource capacity per day.</td>
</tr>
<tr>
<td>ResAssignmentView</td>
<td>This view is based on the table ResAssignment, which stores the activity resource assignments to either project or quotation work breakdown structure (WBS) task.</td>
</tr>
<tr>
<td>ResBookingView</td>
<td>This view is based on the table ResBooking, which stores the activity resource bookings to a project or quotation.</td>
</tr>
</tbody>
</table>

### Changes that will affect tables and fields

These sections contain information regarding code changes to tables and fields that are part of the feature implementation related to Project resource scheduling.

**Resource scheduling**

The table PSASchedEmpResReservation is no longer used to store a resource’s reservations. Instead, reservations are stored in the ResAssignment and ResBooking tables. Both tables use the **Activity resource** field Foreign Key for PSAProjSchedRole.RecId to store a resource’s reservation. The PSAProjSchedRole table is the project team table that has the **Resource** field Foreign Key to ResourceView.RecId and the **ResourceLegalEntity** field Foreign Key to CompanyInfo.RecId to identify which resources are the project’s or quotation’s team members. If the PSAProjSchedRole.Resource field = 0, then this activity resource is a planned resource. A planned resource is a shadow resource that is not backed by an actual resource. PSAProjSchedRole.ResourceCategory is a Foreign Key to PSAProjSchedRole that stores the role of this team member. The ResourceResourceCategorySetup table stores the default time effective resource/role association. However, the resource can be reserved to any role defined by PSAProjSchedRole.ResourceCategory, ignoring the default role definition on the ResourceResourceCategorySetup table. Regarding WBS versioning, the tables ProjPlanVersions and ProjPlanVersionDetails store the WBS tasks versions. Initially, all WBS task data will be stored on these tables while the user is editing the WBS tasks content. After the user clicks the Publish button, the task data will be pushed to the original hierarchy tables (smmActivities and PSAActivitySetup). The resource management feature requires data in the original hierarchy tables and requires a published WBS.

**Price by resource and resource category**
The pricing tables, ProjCostPriceExpense, ProjCostSalesPrice, and ProjRevenueSalesPrice have a Foreign Key from ResourceView.ReclId. The tables ProjHourCostPrice, ProjHourSalesPrice, and ProjTransferPrice have a Foreign Key from ResourceView.ReclId. The field ResourceCategory is part of the Foreign Key for ResourceCategoryView.ReclId. This is done so that pricing is based on a resource instead of a worker and enables pricing setup by resource category.

Common methods to get resource field and lookup resources
There are several resource methods in the ResourceFacade class. The following table includes some of the most common methods.

<table>
<thead>
<tr>
<th>CLASS METHOD</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResourceFacade.findByWorker()</td>
<td>Look up the resource record ID by HcmWorker record ID.</td>
</tr>
<tr>
<td>ResourceFacade.findOrCreateByWorker()</td>
<td>Look up the resource record ID by HcmWorker record ID. If the resource record ID is not found, then a resource record ID will be added to the ResResourceIdentifier table and the resource record ID will be returned.</td>
</tr>
<tr>
<td>ResourceFacade.findByResourceID()</td>
<td>Look up the resource record ID by resource ID.</td>
</tr>
<tr>
<td>ResourceFacade.get...</td>
<td>There are many get methods supported for the resource in the ResourceFacade class. Resource related values like Resource ID, Resource calendar, Resource legal entity, and Resource period can be queried from the ResourceView, ResourceLegalEntityView, ResourceCalendarView, and ResourceSetup tables.</td>
</tr>
</tbody>
</table>

Facade classes
The following table lists the facade classes that you can use as a starting point to interact with the Resource management data model.

<table>
<thead>
<tr>
<th>CLASS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActivityFacade</td>
<td>Contains the methods to retrieve an activity's properties, including ID, Quotation ID, Project ID, Booked/assigned capacity, Remaining capacity, Calendar, Activity number, and Root project activity.</td>
</tr>
<tr>
<td>ActivityResourceFacade</td>
<td>Contains the methods to retrieve an activity's resource properties including Calendar, Name, Resource category, Resource legal entity, Is generic resource, and Is team member.</td>
</tr>
<tr>
<td>PeriodFacade</td>
<td>Contains the methods to retrieve period properties including Start and End date, Period ID, and Period name.</td>
</tr>
<tr>
<td>ResourceCalendarFacade</td>
<td>Contains the methods to retrieve the resource's calendar properties including Calendar data area ID, Calendar ID, Calendar RecID, Capacity, and Dates.</td>
</tr>
<tr>
<td>ResourceCategoryFacade</td>
<td>Contains the methods to retrieve the resource's default resource Category, ID, Name, and Type.</td>
</tr>
<tr>
<td>CLASS</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ResourcePeriodFacade</td>
<td>Contains the methods to retrieve the resource's period date range and update the period method.</td>
</tr>
<tr>
<td>ResourceWorkerFacade</td>
<td>Contains the methods to retrieve the Employment type.</td>
</tr>
</tbody>
</table>
This topic reviews the workflow system in Finance and Operations. It describes the changes that have been implemented since Microsoft Dynamics AX 2012 and also includes links to more information about the workflow system.

The workflow system in Finance and Operations will be familiar to you if you’ve used Dynamics AX 2012. For more information about the workflow subsystem in Dynamics AX 2012, see the following topics.

<table>
<thead>
<tr>
<th>TO LEARN ABOUT THIS SUBJECT</th>
<th>SEE THIS TOPIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>The workflow system</td>
<td><a href="https://technet.microsoft.com/library/dd309672.aspx">https://technet.microsoft.com/library/dd309672.aspx</a></td>
</tr>
<tr>
<td>Workflow participants</td>
<td><a href="https://technet.microsoft.com/library/dd309598.aspx">https://technet.microsoft.com/library/dd309598.aspx</a></td>
</tr>
<tr>
<td>Developing a workflow</td>
<td><a href="https://msdn.microsoft.com/library/cc967389.aspx">https://msdn.microsoft.com/library/cc967389.aspx</a></td>
</tr>
<tr>
<td>Implementing a workflow</td>
<td><a href="https://msdn.microsoft.com/library/cc585061.aspx">https://msdn.microsoft.com/library/cc585061.aspx</a></td>
</tr>
</tbody>
</table>

**Primary changes to the workflow system**

Here are the primary changes that have been implemented in Finance and Operations:

- Integration with the new Application State Machine feature enables workflow events to be bound to state transitions on the underlying entity’s state machine. This binding enables business logic to be centralized within the state machine and also enables the workflow system to be a declarative consumer of that state machine. The workflow metadata can reference a state transition that is performed when a specific workflow event occurs. Therefore, you can do state transitions within a workflow without writing any additional code.

- The workflow editor is now a program that you click one time to download. The editor communicates with Finance and Operations by using services, which means that you can carry forward the rich, graphical workflow design experience from Dynamics AX 2012.

- Workflow development wizards have been ported into Microsoft Visual Studio.

**Additional resources**

*Technical Concepts Guide for Developers*
Transition from Analysis Services cubes to aggregate models

This topic explains how in-memory, real-time aggregate models are used for analytics, and why we transitioned from using Server Analysis Services (SSAS) cubes.

The world is moving to real-time, proactive analytics. Reporting and trending on historical data is being replaced by up-to-the-second visualizations and proactive guidance. In-memory, real-time aggregate models now replace the perspectives that were previously used for analytics.

A historical look at perspectives and cubes

We envision embedded insights playing a key role in the Finance and Operations user experience. This vision has driven us to invest in building analytic capabilities within the product. In Dynamics AX 4.0, we introduced the concept of perspectives. The objective was to present a simpler view of the ERP schema, specifically modeled for reporting. This simpler view was referred to as perspectives. In Dynamics AX 4.0, the system generated reporting models (SMDL models) that enabled you to create ad-hoc reports with SQL Server Report Builder. In Dynamics AX 2009, we added the capability to generate SQL Server Analysis Services (SSAS) projects using metadata definitions in perspectives. These projects become cubes when deployed to an SSAS server. In Dynamics AX 2012, we improved modeling in perspectives and improved tooling support for managing the lifecycle of SSAS projects. You could use Excel, as well as Power View, to explore data and create reports with cubes in Dynamics AX 2012. The SMDL technology was also deprecated. In Dynamics AX 2012, we stopped generating SMDL models.

How perspectives are used now

As a developer, your “contract” with the system was a perspective. The system generated “stuff” to help you achieve your end goal. In Dynamics AX 2012, the “stuff” that was generated was SSAS projects. So the contract between you (the developer) and the system (the BI framework), was as follows:

- You modeled perspectives.
- The system generated the “stuff” needed to enable you to build visuals and reports.

Perspectives are now modeled using add-ins for Visual Studio. (Visual Studio is now the development environment.) Perspectives are comprised of aggregate measurements and aggregate dimensions. As a developer, you have the ability to model simpler schemas for answering business questions using aggregate measurements and aggregate dimensions. Aggregate measurements can be used to define data entities (called aggregate data entities) which can be directly bound to Finance and Operations forms as a data source. Aggregate data entities can also be used to

- Expose data to PowerBI.
- Access data programmatically using the AXQuery object.
Model Aggregate Data entity

Expose to PowerBI via ODataV4

Create query from Data entity

PowerBI Dashboards

Pin a PowerBI tile

Model a chart

Dynamics 365

Aggregate Measurement

"Star schema" modeled in AOT

Create a Query in code

X++ code

New constructs to model expressions and aggregates
In this tutorial, you'll migrate an upgraded Microsoft Dynamics AX 2012 R3 cube schema to the entity store in a Finance and Operations application. You'll use the sales cube that was included in Dynamics AX 2012 R3 as an example.

The entity store will support near real-time Microsoft Power BI integration scenarios, as shown in the following diagram. For an overview of Power BI integration with entity store, see Power BI integration with entity store.

New Power BI features included in the May 2016 and November 2016 updates

This tutorial requires the Dynamics 365 for Operations May 2016 update or later. You will use the following new capabilities in this tutorial:

- Stage an aggregate measurement in the entity store and refresh the data from Dynamics AX. You might prefer this option over in-memory real time aggregate measurements when:
  - You upgrade a Dynamics AX 2012 cube.
  - Your aggregate measurements are very large.
  - Data freshness (latency) from a few minutes up to a few hours is acceptable for reporting.
- Use the batch framework to schedule a recurring refresh. For this release, only a full refresh is enabled.
- Create reports using Power BI desktop in a developer/test environment.
- Leverage the direct query option when creating Power BI content. For example, you can create larger models without relying on OData as the data refresh mechanism.
- Migrate reports from your development environment to a production environment using Lifecycle Services (LCS).
- As a partner or an ISV you can distribute Power BI content as part of an LCS solution to your customers.
- If you're using the November update (platform release 1611) or later, some steps in this document...
Change upgraded aggregate measurement properties

As part of the code upgrade process, analysis services projects from the Application Object Tree (AOT) in Dynamics AX 2012 can be migrated to the new aggregate measurements metadata format.

1. Launch Visual Studio and create a new project in Application Suite.

   **NOTE**
   You can create a model and include the customized aggregate measurement within that model. For more information, see [Customize through extension and overlayering](#).

2. Open Application Explorer. Go to **Analytics > Perspectives > Aggregate measurements**. You will notice a set of aggregate measurements that were upgraded from Dynamics AX 2012 R3, as well as the measurements that ship in the current version.

3. Select **SalesCube**. Right-click and select **Duplicate in project**.

4. An aggregate measurement with the name **SalesCubeCopy** will be added to the project.

5. Rename this measurement. Select **SalesCubeCopy** in Solution Explorer. Right-click and select **Rename**. Enter **SalesCubeV2** as the new name.

6. Double-click **SalesCubeV2** to launch the Aggregate measurement designer. Notice the structure of the aggregate measurement that was migrated from Dynamics AX 2012.

7. The Sales cube in Dynamics AX 2012 encompassed a broad subject area related to Sales. In this case, let's create a smaller, more focused Power BI model using the metadata that was upgraded. Expand the **Sales Order Lines** measure group and review the list of measures and dimension references.

   **NOTE**
   Leveraging the modeling capabilities you can quickly make a few enhancements to this model. Suggestions for improvements:
   - Replace views/tables that have been used to model the measure group (and/or dimensions) with an entity. You can model an entity using the underlying view and replace the view with the corresponding entity. This will enable you to leverage upcoming features such as incremental refresh and security.
   - Remove unwanted dimension references by adding the corresponding field to the attributes node. For example, the Sizes dimension reference can be removed because the **Size** field in the measure group is sufficiently descriptive. This will improve the runtime performance of queries as well as refresh times.

8. Select the **SalesCubeV2** root node in the Aggregate measurement designer. Right-click and select **Properties**.

9. During upgrade, aggregate measurements are set to the legacy property flag, **SSASCube**. You need to change this property to one of two supported usage types. Previously, **InMemoryRealTime** was supported as usage for aggregate measurements. **StagedEntityStore** is supported as a new usage type.

   **NOTE**
   Modify the usage property to **InMemoryRealTime** if you plan to use the Aggregate measurement for embedded BI scenarios as well as Power BI integration. If you are using the Aggregate measurement only for Power BI or Cortana Intelligence Suite integration, select **StagedEntityStore**.
10. Save the project. Right-click the project in Solution Explorer and select **Rebuild**.

11. After the rebuild operation is finished, save the project, and then close Visual Studio. This completes the development work. You will author reports as a report developer or a power user.

**Refresh the entity store**

As an administrator you can configure the refresh of the aggregate measurement using the client.

1. Launch the Dynamics AX client and navigate to **System Administration > Setup > Entity Store**. The **Entity Store** form shows a list of aggregate measurements that are available for deployment to the entity store.

2. Notice that **Sales Cube** (which was upgraded from Dynamics AX 2012) is not available for deployment to the entity store. **SalesCubeV2**, which you created in the previous step, can be deployed to the entity store.

3. Select **SalesCubeV2** from the list, and click the **Refresh** button. The **Refresh** dialog box will display. Expand the **Run in the background** tab.

4. Provide a descriptive name in the **Task description** field. Optionally, you can select the **Recurrence** tab and create a recurring schedule instead of a one-time refresh. Click **OK**.

5. The system will create a batch job for refresh of the aggregate measurement in the entity store.

**Authoring a report on Sales by State with Power BI desktop**

This step requires that you install Power BI desktop tool that can be downloaded from [Microsoft Power BI Desktop](#).

1. Launch Power BI desktop. You may need to apply updates. A welcome page will display. Click **Get data**.

2. Alternatively, when Power BI desktop launches, on the **Home** tab select **Get Data > SQL Server**.

3. In the **SQL Server Database** dialog box, enter the server name and the name of the entity store database. If you deployed a developer environment, you can enter ‘.’ as the server name and **AxDW** as the database name. If you are working in a test environment, you need to get these parameters from your system administrator.

4. Select the **DirectQuery** option. In this exercise, you will create Power BI reports that are executed directly on the entity store. If you had used the **Import** option, Power BI would cache data from the entity store and you would need to periodically refresh the Power BI model. **Import mode is currently not supported with reports written using entity store**. Click **OK**.

5. Next you will see the **Navigator** dialog box. Navigator enables you to select tables and views from the entity store that you want to report on. Enter **Sales** in the search box. The system will filter entities that are related to the **SalesCubeV2** aggregate measurement that was previously created.

   **NOTE**

   The entity store stages the aggregate measurements that have been created. While entities within each aggregate measurement are prefixed and stored as individual tables, Power BI desktop enables you to combine data from multiple aggregate measurements.

6. You will create a report that shows sales by state. Select **SalesCubeV2_Customer** and **SalesCubeV2_CustomerInvoices** from Navigator and click **Load**.

7. You will notice Power BI designer with **Fields** present in the entities that you have chosen (on the far right), as well as available visualization.

**Create a surrogate key that links customers and invoices (applies to Platform versions before November 2016)**
Surrogate keys are generated in aggregate measurements staged into entity store. Power BI desktop does not enable you to relate table joins using multiple fields (also known as, composite keys). The SalesCubeV2_Customer entity does not have a surrogate key (such as AX RecID) defined in it. Next, you will create a surrogate key that enables relating a customer entity to invoices.

1. Select the ellipsis (…) icon next to the SalesCubeV2_CustomerInvoices entity. Right-click and select New Column.

2. Enter the following expression in the Formula editor window.

   \[ \text{FKCustomer} = \text{CONCATENATE(CONCATENATE(SalesCubeV2_CustomerInvoices[DATAAREAID], "-"), SalesCubeV2_CustomerInvoices[ORDERACCOUNT])} \]

3. When completed, your formula should look similar to the following.

4. Notice that a new field, FKCustomer, is shown in the list of fields for the SalesCubeV2_CustomerInvoices table. Because this field is used to relate two tables, you can hide it from end users by right-clicking the field and selecting the Hide option.

5. Next, create a similar field in the SalesCubeV2_Customer table. Select the ellipsis (…) icon next to SalesCubeV2_Customer entity. Right-click and select New Column.

6. Enter the following expression in the Formula editor window.

   \[ \text{FKCustomer} = \text{CONCATENATE(CONCATENATE(SalesCubeV2_Customer[DATAAREAID], "-"), SalesCubeV2_Customer[CUSTOMER])} \]

7. Notice that the field FKCustomer is shown in the list of fields for the SalesCubeV2_Customer table. Because this field is used for relating two tables, you can hide it from end users by right-clicking the field and selecting the Hide option.

Relate invoices and customers
NOTE
You can relate the surrogate keys already created within entity store. If not, you must relate the surrogate keys that you created manually. Next you will create a relationship between SalesCubeV2_CustomerInvoices and SalesCubeV2_Customers entities.

1. Click the Manage Relationships button on the Power BI ribbon. You will see the Manage Relationships dialog box. Click the New button.
2. In the Create Relationship dialog box, select SalesCubeV2CustomerInvoices as the first table in the drop-down list. Scroll to the right and select the FKCustomer field as the column to relate to.
3. In the second drop-down list select SalesCubeV2Customer as the table. Scroll to the right and select FKCustomer as the column to relate to.
4. Select the Make this relationship active option if it is not already selected. Click OK to continue.
5. You will notice the newly created relationship in the Manage Relationships dialog box. Click the Close button.

Create a Sales by state report
1. To create a report that shows sales by customer group, drag the CustomerInvoiceAmountAccountingCurrency field from the SalesCubeV2_CustomerInvoices table and drop it on the Power BI desktop canvas. Next, drag the CustomerGroupName field in the SalesCubeV2_Customer table to the same grid.
2. Change the chart type to a doughnut chart. You should see a report similar to the following.

3. You can create additional visuals using the Power BI desktop. When you save, you will notice that the file has a PBIX extension.
4. Save the report to your desktop.
5. At this point the report is fully functional (with data from your environment) and you can continue to use the Power BI desktop or upload this report to PowerBI.com and continue with data exploration.
6. Next, you will migrate this report to a production environment using LCS so that you can see this report with production data and share it with other users.

**Publish the report and the model**

Publishing a report and model requires uploading the report to Lifecycle Services, migrating the aggregate measurement to your production environment, configuring the client to point to the correct LCS library, and publishing your reports in your production environment.

**Upload the report to Lifecycle Services**

Microsoft Dynamics Lifecycle Services (LCS) is the tool used to migrate development artifacts from developer to production environments. In the May 2016 update, LCS supports migrating PBIX files (authored using the entity store) between environments.

1. Open LCS from the developer environment. If you haven’t created a project in the LCS environment, create a project.
2. Scroll to the right and you will notice the **Asset Library** icon. Click the icon and launch **Asset Library**.

Notice that the asset library enables adding **PowerBI report models** (PBIX files) as implementation artifacts to a project.

1. Select the plus (+) icon to add a new asset.
2. Provide a name and a description. Click **Upload** and then locate the file that you saved in an earlier step.
3. After you successfully upload the file, click **Confirm**. Notice that the file is uploaded into LCS as an implementation asset. LCS supports managing versions and releases for Power BI reports. You can maintain several versions and publish reports to other environments, just as you would for other implementation artifacts. Because you added the PBIX files as an asset within an LCS project, environments that you deployed using that project will have access to this report.
4. Optionally, you can publish this report so that all of your projects can access the shared assets. If you are a partner or an ISV, and want to share this report with your customers, you would share this asset to your global library and enable your customers to import the asset into their respective LCS projects. To do this, select the **Save to my library** option.

**Migrate the aggregate measurement to a production environment**

1. You need to migrate the aggregate measurement that you modified in the developer environment to the production environment. You can follow the instructions in Generate a deployable package. create-apply-deployable-package.md.
2. After you successfully publish the model, perform the steps outlined in the **Refresh the entity store** section of this tutorial, so that the entity store is updated with data.

**Configure an LCS project**

If you haven’t already done so, associate your environment with an LCS project so that Finance and Operations apps can consume assets within the project.

1. Launch the client from the instance that you want to use to deploy the Power BI reports. Typically this is the test or a production instance where you want to see a report with a different set of data than what you worked with as a report developer.
2. Open **System Administration > Setup > System parameters**. Select the **Help** tab. Using the **Lifecycle services help configuration** list box, select the LCS project that you uploaded the PBIX file to. Click **Save**.
Publish Power BI reports to a production environment

1. Open System Administration > Setup > Deploy PowerBI from the client. You will see the file that you uploaded to LCS.

2. Select the Sales Report file and select the Deploy Power BI files option on the menu bar.

   **NOTE**
   You may be asked to consent publishing to the PowerBI.com service. Click the link to provide consent. When consent is complete, you need to go back to the original browser window and click the Close button.

3. After you successfully publish the file, the Power BI report will appear in your PowerBI.com subscription. You will notice that the report now points to the entity store in the production environment.

Continuing with PowerBI.com

As an administrator or a power user, you have successfully authored and published a Power BI report to the production environment using the entity store. You can perform several additional steps using Power BI functionality.

- Optionally, you can apply record-level security to the dataset to restrict users from seeing data they are not allowed to view in Power BI.
- You can create an organizational content pack and share it among users in a group.
  - You can export datasets, reports, and dashboards from your PowerBI.com instance as a new content pack to a selected group of users.
  - Note that organizational content packs adhere to any record-level security rules that you defined at the dataset level.
- Users can personalize their workspaces by adding Power BI tiles or reports.

Additional resources

Model aggregate data
There are significant differences in budget planning between Microsoft Dynamics AX 2012 and Dynamics 365 Finance. Some features were not upgraded and therefore require reconfiguration. This topic explains what must be reconfigured and also describes new features that should be considered after the upgrade is completed.

Budget planning in Finance has many enhancements that weren't available in Dynamics AX 2012. This topic explains the changes that customers who upgrade must make. It also points out the new features that should be considered in the upgrade process. Because of the extent of the changes, any existing budget plans will not be able to be opened until the changes that are outlined in this topic are made. However, reports should continue to work and not require additional changes.

Overview of changes

Many significant changes have been made in Budgeting for Finance and Operations. These changes are intended to make Budget planning easier to configure and more reusable, to reduce year-over-year maintenance and setup. The following areas in AX 2012 no longer exist in Finance:

- Budget plan templates (Budget planning configuration)
- Budget plan folders (Budget planning configuration)
- Scenario constraints (Budget planning configuration)
- Templates for Budget planning stage rules and templates (Budget planning process)
- Matrix fields for worksheet templates
- Budget plan Microsoft Excel template wizard

Some new concepts can't be directly upgraded from the previous functionality. Therefore, you must complete some reconfiguration to address these new concepts. The following sections describe the concepts that have replaced the items in the preceding list.

Columns

Columns are a new concept that replace parts of the Excel template and also matrix fields. Columns can represent a period, month, quarter, year, or all time. The time reference is dynamic. It points to a relative period or year in reference to the budget process. For example, a Prior Year January column references fiscal period 1 for year -1. A column is specific to a budget plan scenario, such as actuals or budget request.

Layouts

Layouts are a new concept that replace the Excel template. Layouts contain the columns that define which budget or actuals data and periods should be shown. Layouts are also shared between the client and the Excel add-in. Therefore, the user experience when you enter or view data in the Finance and Operations client is better than the user experience in AX 2012. To enter data in the Finance client, you're no longer limited to viewing and entering a single scenario in a transaction view. Instead, a comparison view lets you easily view and enter amounts for multiple periods and accounts at the same time. Layouts can also be defined so that you can enter and view currency, comments, and other optional data. Layouts also let you define which ledger dimensions and dimension descriptions should be shown. Layouts also incorporate scenario constraints to define which columns in a template can be edited and which columns should be available in Excel. After you define a layout, a template is generated for it. This template, in turn, creates the corresponding Excel template. You can then edit the Excel template to incorporate more formulas and formatting, and then upload it again. Layouts are then assigned to each stage rule on the Budget planning process page. Therefore, the layouts replace templates, which were assigned and used in a similar manner.
Budget planning processes

Budget planning processes are mostly the same as in AX 2012. The most significant change is the replacement of templates with layouts. If any processes were previously completed in AX 2012, the processes are updated to a status of in-progress so that changes can be made. You must assign layouts will need for each stage rule to determine which scenarios and time periods appear when the plan is opened in the client. The layouts also determine which Excel template is opened outside Dynamic 365 Finance so that you can view the budget. 

**Default account structure** is a new required field for the Budget planning process. For each Budget planning process, assign the primary account structure that should be used for budgeting.

Attachments

In AX 2012, justification documents were saved to an attachment folder. No previous justification documents are upgraded. Justification documents are now stored in the database. If this information should be saved in the upgraded version, you can upload final justification documents for each plan as an attachment by using the Justification button on the Action Pane. In AX 2012, Excel worksheets for each budget plan were created based on the template. In Finance, all plans open a copy of the layout. However, no changes to the Excel file are saved. Any formulas or supporting information that were used on a per-plan basis must be added via comments, a justification document, or some other supplemental process.

Configuring an upgraded environment from AX 2012

To help you determine how to configure the upgraded system, the following example uses an upgraded budget process from AX 2012 demo data. Default configuration data for columns were created to help with the upgrade process. You can update or delete this default data if doesn’t meet your configuration requirements. **Note:** There are new required fields that won’t be set in the system. If you get stuck on a page, such as the Budget planning configuration page, and can’t navigate away, you can close your browser and then reopen it to a different page to enter details in the correct order. There are required fields that aren’t yet set. Therefore, issues might occur until everything is configured and all required fields have been set. This topic explains how to set these fields, as required. Here are some of these required fields:

- **Budget planning process** page: Default account structure field
- **Budget planning process** page: Layout field on the Budget planning stage rules and layouts FastTab

Define columns and layouts

1. On the Budget planning configuration page, click the Columns tab. As part of the upgrade, new columns are automatically created based on your budget plan lines. Columns now use dynamic dates, where the time and year are offset from the fiscal year that is defined in the Budget planning process. **Note:** For performance reasons during upgrade, it’s assumed that all budget cycles represent calendar years, not fiscal years. If you use fiscal years, you must make edits to correctly map the columns to their fiscal year. For example, the following elements existed in AX 2012:

- **Budget plan scenarios:** Actuals, Baseline, Budget Request, Budget Approved
- **Budget plan lines for all scenarios in 2017, and Actuals for both 2017 and 2016**

The following columns will be created in Finance and Operations:

<table>
<thead>
<tr>
<th>COLUMN NAME</th>
<th>BUDGET PLAN SCENARIO</th>
<th>COLUMN TIME PERIOD</th>
<th>YEAR OFFSET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan Scenario 1</td>
<td>Actuals</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Jan Scenario 2</td>
<td>Baseline</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Jan Scenario 3</td>
<td>Budget Request</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Jan Scenario 4</td>
<td>Budget Approved</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Alternate layouts are a new feature that lets you view plans in different layouts. One or more layouts can be
Budget milestones

As part of the budget process, it’s vital that you understand key dates and deadlines. You can now configure dates so that they have descriptions. Budgeting users will see these descriptions when they open budgets to edit or view anything that is assigned to them.

Copy from Budget Plan allocation

A new allocation method lets you distribute from a parent plan to a child plan without having to go through an intermediate level in the hierarchy. This method is especially useful for customers who previously created financial dimension just for budget distribution and approvals.

Generating budget plans from new budget sources

The following options were added as periodic processes. These options let you generate a budget plan by using existing data from another module as the starting point:

- Generate Budget Plan from Demand Forecast
- Generate Budget Plan from Supply Forecast
- Generate Budget Plan from Project
- Generate Budget Plan from Budget Register

More complete tracking of amounts

In AX 2012, budget planning had a single plan amount that was stored for each value. In Finance, the data model has been expanded. There are now accounting currency, transaction currency, and reporting currency amounts for each value. During the upgrade, these new columns are automatically filled in for existing data.

Do not convert currency in aggregation

Typically, when a child plan is aggregated to a parent level, the amounts are automatically converted from the transaction currency to the accounting currency for the organization. When you set the Do not convert currency in aggregation option to No the aggregated amounts remain in the original currency. Therefore, this option allows for more accurate adjustments that are affected by exchange rate fluctuations.

Looking back from a budget plan to other modules that contributed to the budget

Budget plans can be generated from demand or supply forecasts, project, and other areas. The Budget plans by dimension set inquiry includes several options that let you run queries to identify the data that was the source for the budget plan.

Overwrite or append to plan for allocation schedules

If there are multiple sources for amounts that must be distributed, you can specify that the amounts should be additive. In this case, the amounts don’t overwrite any existing amounts. Instead, they are appended to the existing amounts.

Default financial dimension set for budget planning configuration

The Budget planning configuration page now includes a field where you can specify the default financial dimension set. Although this field is an optional field, it might be required for certain inquiries. It might also be required if you want to group or filter reports grouping by dimension set.

Data entities

Several data entities have been added to enable rapid implementation of Budget planning. The entities also let you make many changes through Excel. Therefore, you don’t have to create items one at a time through the client. Here is a list of the new data entities:

- Entity name
- Budget parameters
- Budget plan parameter
- Budget plan scenarios
- Budget plan stages
- Budget plan workflow stage
- Budget plan allocation schedules
- Budget plan stage allocations
- Budget plan priorities
- Budget plan columns
- Budget plan layout elements
You can use the Microsoft Dynamics AX 2009 Data migration tool (DMT) to migrate your data from AX 2009 to Finance and Operations. Using the DMT is the only supported upgrade path from AX 2009. The DMT helps you find and fill gaps between the table schemas for each version, as well as helping you move your data.

**NOTE**

Start your cloud migration journey with a no-charge, no-obligation migration assessment though the Dynamics 365 Migration Program.

**Architecture**

The following illustration describes the architecture of the DMT, and how data from the source system (AX 2009) is processed and moved to the target system (Finance and Operations).
Data migration process

The following illustration shows the overall process of collecting and preparing the data in your AX 2009 instance and then importing that data into your new environment.

Before you can use the DMT to export data from the source environment (AX 2009), you must complete the following pre-processing tasks:

- Mapping the table fields between the source and target environments
- Applying conversions to the source data
- Setting up default values for the source data
- Applying query filters

Because there can be multiple legal entities in the source system, you must select the legal entities that contain
the data to migrate. For the selected legal entities, you can review the source tables and their row counts. You can also view any virtual companies. Finally, you can analyze virtual companies that legal entities are attached to and the related tables.

Successful migration of exported data requires that a source table be mapped to an equivalent target data entity. You set up the mapping by using a Microsoft Excel mapping file that allows for automatic mapping of the source and target fields of each table. The mapping file also includes the data from the schema of the new data entities and the default data that is required in some tables.

Before you can migrate data from AX 2009, you must complete the following tasks to meet the migration requirements:

- Select target dimensions that correspond to the source dimensions that are populated based on the selected legal entities.  
- Review the inventory dimensions that are included with the selected legal entities.  
- Select the chart of accounts for each legal entity, or consolidate multiple legal entities into a single chart of accounts.  
- Complete the basic ledger setup.  
- Apply data conversions to the source data, based on the extended data type (EDT) of the field.
This topic explains how to set up the Data migration tool (DMT) so that you can migrate data from Microsoft Dynamics AX 2009 to Finance and Operations.

IMPORTANT
At this time, the DMT is in private preview. If you are interested you can sign up for the Preview Program. The public release date for the DMT has not been set.

Prerequisites
- The Microsoft .NET Framework version 4.5 or later.
- Microsoft SQL Server machine that has Microsoft SQL 2012 Native Client installed.
- The Microsoft SQL Server Integration Services (SSIS) service is installed and running on the machine where the DMT service will be installed.
- SQL Server authentication must support both SQL authentication and Microsoft Windows authentication.
- Microsoft Access database engines that follows the version guidance in the following table.

<table>
<thead>
<tr>
<th>OFFICE VERSION</th>
<th>SQL SERVER 2008</th>
<th>SQL SERVER 2012 AND LATER</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Microsoft Office on the VM</td>
<td>Access engine 32-bit</td>
<td>Access engine 64-bit</td>
</tr>
<tr>
<td>Microsoft Office 32-bit</td>
<td>Access engine 32-bit</td>
<td>Access engine 64-bit</td>
</tr>
<tr>
<td>Microsoft Office 64-bit</td>
<td>Access engine 32-bit and 64-bit</td>
<td>Access engine 64-bit</td>
</tr>
</tbody>
</table>

- Microsoft Dynamics AX 2009 SP1 5.0.1000.52 or later.
- The prerequisite patch (axpatch.exe) installed. To find the patch, from the location where you downloaded and extracted the zip file, go to <pre-requisiteforpatch><application>.

Install DIXF service

1. Go to the location where you extracted the zip file, and then, in the DIXF msi folder, right-click DIXF_Service_x64.msi, and select Run.
2. When the wizard starts, select Next.
3. Accept the license terms, and then select Next.
4. Select an account for the service, and then select Next. The account should have admin rights. If you select the Network Service check box, verify that the network service account has admin rights. Otherwise, clear the check box, and enter an admin account user name and password. Then select Next.
5. Select the SQL Server version, and then select Next.
6. Select Install, and then, when the wizard is completed, select Finish.
Copy binaries

Go to the location where you extracted the zip file, and copy the following files to the Program Files (x86)\Microsoft Dynamics AX\50\Client\Bin folder:

- Microsoft.Dynamics.AX.Framework.Tools.DMT.dll
- Interop.Shell32.dll

Install DMT components for AX 2009

There are two ways to install the DMT. You can use the combined XPO file or an application hotfix. If you're using a Microsoft Dynamics Lifecycle Services (LCS) Implementation project, use the application hotfix. Installation takes approximately seven hours.

**Combined XPO file**

1. Extract the combined XPO file from DMT_V1.0\CombinedXPO.
2. Import the combined XPO file into AX 2009.
3. Copy the label file from DMT_V1.0\Label file to the Program Files\Microsoft Dynamics AX\50\Application\Appl\<NameOfYourDeployment> folder.
4. Restart the Application Object Server (AOS) instance.
5. In AX 2009, select Data migration > Setup > Compile and synchronize DMT application.

Note that the combined XPO file is imported into the layer that the user is signed in to.

**Application hotfix**

1. Go to DMT_V1.0\ApplicationHotfix\DynamicsAX2009-KB4010403-SP1, right-click setup.exe, and then select Run.
2. In AX 2009, in the Application Object Tree (AOT), notice that the LegalEntityId field has been added to the DMTCustomerAddressView and DMTVendorAddressView views.
3. Select Data migration > Setup > Compile and synchronize DMT application.

Parameter setup

Go to the location where you extracted the zip file, and find defaultvalue.xlsx.

**NOTE**

The file is saved in .xlsx format. Don't change the extension. When you provide this file as input for the Default configuration parameter, select All Files so that you can select the .xlsx format. If you don't select this format, errors will occur when you start to generate mappings.

1. In AX 2009, select Data migration > Setup > Configure default maps, and enter the appropriate information in the following fields:
   - Default configuration – Enter the path of the Microsoft Excel file.
   - Export file path – Enter the server path that can be accessed by the service.
   - SQL Server user and password – Enter the SQL authentication credentials for the AX 2009 database.
2. Close the form.

3. Under Setup, select Configure connections, and enter the appropriate information on the following fields:
   - DIXF service host – Enter the host name of the DIXF service installation.
Multi-box setup

For a multi-box setup, you must have the following machines:

- Machine A, where the AX 2009 database and DIXF service are installed
- Machine B, where the AX 2009 AOS instance is installed
- Machine C, where the AX 2009 client is installed

In this three-machine setup, machine C is configured to connect to the AOS instance on machine B. Machine B is connected to the database that is configured on machine A.

DIXF service machine prerequisites (machine A)
The DIXF service on machine A has the following prerequisites:

- SQL Server 2008/2012/2014
- The .NET Framework version 4.5
- Access database engines
  - For SQL Server 2008: Access engine 32-bit and 64-bit (if Microsoft Excel is 64-bit)
  - For SQL Server 2012 or later: Access engine 64-bit
- AX 2009 database (configured on SQL Server)

AOS machine prerequisites (machine B)
The AOS installation on machine B has the following prerequisites:

- AX 2009 AOS Server
- Application files

Client machine prerequisites (machine C)
The client installation on machine C has the following prerequisites:

- AX 2009 client

Shared folder permissions
The path of the default configuration file and the export package file should be shared, and client users and the DIXF service should have read/write access to these files. To grant this access, select Data migration > Setup > Configure and generate maps, and then select Validate path to verify that the required access is available.

Set up parameters
1. Select Data migration > Setup > Configure connections.
2. In the DIXF service host field, enter the name of the remote machine where the DIXF service is installed. By default, the name is localhost.
3. Select Validate to validate that the client can access the DIXF service.

Workarounds
If you receive an error message that states, "DIXF service is unavailable," complete the following workaround to enable a service connection for port 7000.
1. Open port 7000, and then, for inbound rules on the DMT service machine, select **Firewall settings**, and then select **Run > wf.msc**.
2. Select **Inbound Rules > New rule**, and then, on the **Rule Type** tab, select **Port**, and then select **Next**.
3. In the **Specific local ports** field, enter **7000**, and then select **Next**.
4. Select **Allow the connection**, and then select **Next**.
5. Select all three check boxes to apply all the rules, and then select **Next**.
6. Enter the name of the rule, and then select **Finish**.
7. Repeat these steps for outbound rules.
Before you can migrate your data from Microsoft Dynamics AX 2009 to Finance and Operations, you must align your source data with your target environment. This topic explains how to generate source-to-target mappings.

Before you can generate maps, you must provide the target URL, tenant URL, and service app ID to validate the connection.

**NOTE**

When you create a new app under Microsoft Azure Active Directory (Azure AD) in the Azure portal, you have two options, **Web API** and **Native**. Select **Native**, and grant permissions to the native Azure AD app.

Prerequisites

Before you generate the data maps between the source and target environments, you must install the Data migration tool (DMT). For more information, see [AX 2009 migration - Install the Data migration tool](#).

Generate maps

Follow these steps to generate maps for data migration.

1. In AX 2009, in the navigation pane, go to **Data migration** > **Setup** > **Configure connections**.
2. Review the field information to verify that it's correct, and then click **Validate**.
3. After the validation is completed, close the form.
4. Under **Setup**, click **Configure and generate maps**.
5. Verify that the information in the form is correct, and then click **Validate path**.
6. After validation is completed, click **Generate maps**.
Packages are created by following a predefined sequence. This sequence is based on the dependencies that the data entities have on each another. Because of these dependencies, when you import data entities, you must import the data entities in the defined order. Otherwise, you might encounter issues during import and configuration.

The Data migration tool (DMT) provides twenty predefined templates, as shown in the following illustration.

You can customize an existing template, or you can create your own templates as you require.

Follow these steps to view and select the entity lists that will be used in the templates for migration.

1. In Microsoft Dynamics AX 2009, click Data migration > Common forms > Entity list, and then click Apply sequence. Close the message box.
2. Verify that the correct legal entity is selected, and then, in the Show field, select to view either all entities or only those entities that should be considered for migration.
3. In the Template name field, select a template.
4. In the Module selected pane, select the module that contains the data entities to migrate.
5. On the Entity details tab, select the Select for migration check box for every entity line that you want to migrate.
6. Click Apply sequence.
7. To create a customized template, in the Application Object Tree, go to Resources, and create a new template in XML format.
When you create a definition for migration, you determine which entities should be packaged and exported together, and then put all the entities together in a migration group. A migration group is a set of entities that must be processed in a sequence, or that can logically be grouped together. The entities in a migration group are exported together, either from the source to staging or directly to a file package. In a migration group, you also associate legal entities. Migration groups must be set up before you begin the export process.

Follow these steps to create a migration group.

1. In Microsoft Dynamics AX 2009, in the navigation pane, click **Data Migration > Common forms > Create migration group**.
2. In the **Migration group** form, press CTRL+N or click **New** to create a new migration group.
3. Enter a name for the migration group. Then press Tab to move to the **Company** field, and click **Select company**.
4. In the **Select company accounts** form, select one or more companies to add to the migration group, and then click **OK**.
5. In the **Migration group** form, click **Entity**, and select the lines to include in the migration.
6. Fill in any gaps in the field mapping, as required.
7. Click **Apply sequence**, and then close the form.
You can use the Data Import/Export Framework (DIXF) service in Microsoft Dynamics AX 2009 to retrieve data that must be migrated to Finance and Operations. The export process is completed through a job ID. When you export, you can specify how the export job is defined. You can select the source data to export, the conversion value, and the field mapping. You can also apply a query to each source to limit what is exported.

The export package that the Data migration tool (DMT) generates can consist of one or many data entities. A typical data package consists of a group of entities for a specific task, such as import. For example, the data entities that are required for system setup might be part of one data package. The format of a data package is a compressed file that contains a package manifest, a package header, and any additional files for the data entities that are included.

Before you create a data package, plan out what should be included. In this way, you help guarantee that the correct entities, entity sequence, and fields are included.

Follow these steps to export the data package.

1. In AX 2009, in the navigation pane, click Data migration > Common > Create migration group.
2. In the Migration group form, select the migration group to export, and then click Export now.
3. In the Export data form, update the export file path as required, and then click OK.
Data can be imported for a group of logically related entities that are sequenced in the correct order. You have three options for importing Microsoft Dynamics AX 2009 data that you want to migrate:

- AX 2009
- Finance and Operations

**AX 2009**

You can import data for migration directly from the source system. Follow these steps.

1. In AX 2009, in the navigation pane, click **Data migration**.
2. Go to **Common > Create migration group**.
3. In the **Migration group** form, select the migration group to export, and then click **Export now**.
4. In the **Export data** form, select the **Import package in target** check box, and then click **OK**.

**Finance and Operations**

You can import data for migration by using your Finance and Operations environment. Follow these steps.

1. Sign in to your environment by using an Administrator role.
2. On the dashboard, select the **Data Management** workspace.
3. Select **Import**.
4. Enter the name of the package, and then, in the **Source data format** field, select **Package**.
5. Select **Upload**, and then select the appropriate package file from the location for the data that is being imported. All the files from the package are imported.
Migrate your code

To migrate your code from Dynamics AX 2012 to Dynamics 365 Finance, Supply Chain Management, or Commerce, use the "Migrate and Create Solutions" methodology in Lifecycle Services.

Key concepts

The following links (also included in the methodology) describe key concepts and steps in the migration process. The links are listed here in the order that we recommend you read them.

- Prepare to migrate code to Finance and Operations
- Model split
- Removed or deprecated features for Finance and Operations
- Deprecated APIs

Learning path

- Upgrade Dynamics AX 2012 to Finance and Operations apps

Additional concepts

- Solve dependencies among models by using delegates during code migration
- How to import a SQL Server Analysis Services Project into the AOT
- Upgrades, updates, and hotfixes resources
- Workflow subsystem updates in Finance and Operations
- Migrate upgraded AX 2012 R3 sales cubes to the entity store
This topic describes how the Lifecycle Services code upgrade service and Visual Studio tools help you migrate your code and metadata from Dynamics AX 2012 R3 to Finance and Operations. Most of these steps also apply to code migration between two major versions of Finance and Operations.

Prerequisites

You will need access to a Finance and Operations development environment using Remote Desktop, and be provisioned as an administrator on the instance. We recommend you become familiar with some of the Finance and Operations development, customization, and user interface concepts before you upgrade your code. Here are some references.

- Development tools
- Models and packages
- X++ programming language
- Extensions and Overlayering
- User interface development

Overview of the code migration process

Model split

The Finance and Operations application is split into several packages, or assemblies:

**Platform Packages**

- Application Platform
- Application Foundation
- Test Essentials

**Application Packages**

- Application Suite
- Other application packages.

ISV and customer code that is migrated from Dynamics AX 2012 R3 will be re-baselined into the correct package.

Auto-migration using the LCS Code Upgrade service

The LCS code upgrade service takes a Dynamics AX 2012 R3 model store as input and completes the following tasks:

- Converts metadata into the latest format.
- Re-baselines metadata, by moving and merging, into the right model.
- Provides an estimation to understand the effort required to upgrade the solution.
- Runs migration rules that auto-migrate parts of a solution.
- Runs migration rules that inform developers what to manually fix by using TODOs.
- Automatically checks-in the upgraded solution into your Azure DevOps project.

To configure and run the code upgrade service, see [Configure the code upgrade service in Lifecycle Services](#)
Manual migration steps

After you upgrade your code using the LCS code upgrade service configure your developer VM and Azure DevOps to connect to the upgraded code branch.

- Configure one-box development environments
- Configure the Azure DevOps mapping during code migration

The code upgrade service will provide with Visual Studio solutions that you can open to compile your code. A code merge solution for all elements that contain conflicts and an upgraded solutions for all your upgraded elements. Typically, you can compile the application by fixing compilation errors in the order shown below. The order is determined based on the package dependencies graph, start with the lowest package in the graph. To determine package dependencies, see Models and packages. A typical order is Application Platform, Application Foundation, Directory, ...etc., Application Suite. For each of your upgraded models:

- Fix merge conflicts.
- Fix compilation errors related to a model split (references across packages).
  - Typical error messages are:
    - `<Element Type> X refers to <Element Type> Y which does not exist.`
    - The name `<Name>` does not denote a class, a table or an extended data type.
    - For example, your overlayering customizations may be referencing elements or code that are higher in the package dependency graph:
      - A method in the Directory model is referencing a table in the Application Suite package.
      - A form in the Directory package is referencing a data source in the Application Suite package.
    - You will have to refactor your code to address these dependencies by moving model elements or business logic to higher level packages.
    - Solve dependencies among models by using delegates during code migration describes how to use delegates to solve some of these issues.
- Fix compilation errors.

After you have resolved all of the compilation errors, all packages will compile. Next, you must complete the following tasks:

1. Address guided code upgrade TODOs and code upgrade-specific best practice warnings. Some examples and details are in the sections below.
2. Replace deprecated controls, for example, ActiveX or find an alternative.
3. Apply form patterns and sub patterns to all forms.
4. Validate that all scenarios work in multiple browsers with different sizes for custom patterns.
5. Write and run tests.

Best practice setup

In the Best Practice framework, there is a subset of Best Practice warnings that need to be resolved to complete migration. This applies if you are migrating from Dynamics AX 2012 R3 or earlier.

1. In Visual Studio, click Dynamics 365 > Options > Best Practices.
2. In the Model drop-down menu, select Application Suite (Repeat with all models you are working on)

These rules should be set to “ON” while migrating your solution. The setting is driven by an XML file in the AxRuleSet folder. For example, see the Application Suite xml file, BPRules.xml, located under C:\Packages\ApplicationSuite\Foundation\AxRuleSet.
Debugging

To complete the migration, you need to fix all migration-specific Best Practice rules. The errors will show up in the error list as warnings. In the error list, you will see compiler warnings and best practice errors. Best Practice errors are prefixed with the text BP. For example, BPErrorFormControlPatternUnspecified.

To address code migration tasks, you need to fix all migration-specific Best Practice rules. The errors will show up in the error list as warnings. In the error list, you will see compiler warnings and best practice errors. Best Practice errors are prefixed with the text BP. For example, BPErrorFormControlPatternUnspecified.

Debugging

By default, Finance and Operations optimizes the debugging experience for the files that you are working on. As a result, when you step into a file (F11) that is not in your project, the PDBs are not loaded and you can't debug the code. To work around this, change the project debugging setting by clicking Dynamics 365 ➤ Dynamics 365 ➤ Options ➤ Debugging. Verify that the Load symbols only for items in the solution check box is not selected. This option is selected by default because it improves the debugger speed significantly. Another debugging setting that you may want to turn off is Intellitrace. Intellitrace collects the complete execution history of an application. It creates a lot of noise in the IDE when debugging. To turn off Intellitrace, click Options ➤ IntelliTrace ➤ Enable IntelliTrace, clear the check box, and then click OK. Note that Intellitrace is only available in the Enterprise version of Visual Studio.

Address code migration tasks

When metadata is migrated to Finance and Operations, multiple auto-upgrade scripts are run. In the case where developers need to complete manual migration tasks, TO DOs and Best Practices (BP) have been added.

- TO DOs are prefixed with /* TODO: (Code Upgrade)*/, and need to be fixed as a part of code migration.
- BP migration specific rules also need to be fixed as part of code migration.

This example below uses the PurchCommitment_PSN form to walk you through the migration task of fixing navigation. Specifically, you will see examples of duplicate buttons and Action Pane TODOs.

Setup

1. In Visual Studio, open Application Explorer, and search for the form, PurchCommitment_PSN.
2. Click OK.
3. Right-click the project and select Properties.
4. In the Model property, select Application Suite.
5. In the Company property, select FRSI.
6. Note: The form is located in the French demo data company FRSI.
7. Press Ctrl+F5 to see the form.

While the form looks complete, there are still code migration tasks necessary to be migration-complete.

**Navigation migration tasks**

1. In Visual Studio, build the project, and then on the toolbar, click View > Task List.
2. Click the Comments drop-down list to view the TO DO: (Code Upgrade) tasks.
3. In the list, find the ActionPane TODOs.

**Code upgrade rule - Action Pane**

In Finance and Operations, the following core actions are provided as system-defined buttons:

- New
- Delete
- Edit
- Export

As part of the auto-migration, the Action Pane rule is run to identify redundant buttons. To complete this part of migration, you need to manually:

- Remove or move the code.
- Delete redundant controls in the application code.
In the section below, we will provide examples of how to migrate and modify the code on modeled buttons that replicate system-defined buttons. However, in practice, before making changes similar to those made in this article, the code must first be evaluated with respect to the scenario to determine if it is still needed. First, fix the TODO for the DeleteCmdButton, which duplicates the system-defined Delete button.

1. In Visual Studio, find the TODO shown below, and then double-click the TODO.

   ```csharp
   // Delete button
   /* TODO: (Code Upgrade) [Action Pane Rule] Please consider moving all references to the form task override method and remove the control: DeleteCmdButton */
   deleteCmdButton.enabled(purchCommitmentHeader && purchCommitmentHeader.canDelete());
   PurchCommitmentHeader_DS.allowDelete(purchCommitmentHeader && purchCommitmentHeader.canDelete());
   
   The state of the system-defined Delete button is controlled by the AllowDelete property on the firstmaster datasource. By setting AllowDelete to false, the delete task is kept from executing when the keyboard shortcut is used.
   
2. Replace the TODO and the line of code as shown below.

   - The state of the system-defined Delete button is controlled by the AllowDelete property on the firstmaster datasource. By setting AllowDelete to false, the delete task is kept from executing when the keyboard shortcut is used.

   ```csharp
   deleteCmdButton.enabled(purchCommitmentHeader && purchCommitmentHeader.canDelete());
   PurchCommitmentHeader_DS.allowDelete(purchCommitmentHeader && purchCommitmentHeader.canDelete());
   
   3. In the editor, find and remove DeleteCmdButton from the form design.

   4. Press Ctrl+S to save the form.

   - Next, we will focus on the EditCmdButton that duplicates the system Edit button, handing the two TODOs associated with this button as well as removing this button.

5. In Visual Studio, find the TODO shown below, and then double-click the TODO.

   ```csharp
   /* TODO: (Code Upgrade) [Action Pane Rule] Please consider moving all references to the form task override method and remove the control: EditCmdButton */
   editCmdButton.enabled(purchCommitmentHeader && purchCommitmentHeader.canEdit());
   PurchCommitmentHeader_DS.canEdit(purchCommitmentHeader && purchCommitmentHeader.canEdit());
   
   The visibility of the system-defined Edit button is controlled by the View/Edit mode of the form. By setting CanEdit to false, the edit task is kept from executing when the keyboard shortcut is used.

6. Because the visibility of the Edit button is controlled by the View/Edit mode of the form, you will need to modify this code so it sets that property. Replace the TODO and the line of code as shown in the following graphic.
/* TODO: (Code Upgrade) [Action Pane Rule] Please consider moving all references to the form task override method and remove the control: EditCmdButton */
editCmdButton.enabled(purchCommitmentHeader && isInDraftOrUnderRevisionStatus && !isInWorkFlowReviewState && !isLineReferenced);

if(purchCommitmentHeader && isInDraftOrUnderRevisionStatus && !isInWorkFlowReviewState && !isLineReferenced)
{
    element.design().ViewEditMode(ViewEditMode::Auto);
}
else
{
    element.design().ViewEditMode(ViewEditMode::View);
}

7. Double-click the other TODO for this button.

8. Inspect the code on the modeled Edit button. This logic will need to be moved to the form's task() method.

9. On the left side of the Visual Studio designer, right-click Methods > Override, and select Task, to add an override for the form's Task method.

10. Update the task method as shown below so that the code from above is triggered when the system-defined Edit button is clicked.
public int task(int _taskId)
{
    #Task
    int ret;

    switch (_taskId)
    {
    case #taskEditRecord:

        if (purchCommitmentHeader.WorkflowApprovalState == PurchCommitmentWorkflowApprovalState_PSN::Approved)
        {
            if (Box::yesNo(strFmt("@SPS2140", purchCommitmentHeader.CommitmentNumber), DialogButton::No) == DialogButton::Yes)
            {
                ret = super(_taskId);

                PurchCommitmentHeader_PSN::setWorkflowState(purchCommitmentHeader.RecId, PurchCommitmentWorkflowApprovalState_PSN::NotSubmitted);
            }
        }
    else
    {
        ret = super(_taskId);
    }

    break;

    default:
    {
        ret = super(_taskId);
    }

    break;

    return ret;
    
}

11. In the Editor, find and remove the EditCmdButton from the form design.

12. Press Ctrl+S to save the form.

13. Press Ctrl+F5 to view the form. Notice the Delete and Edit buttons in the Commitment tab have been removed.

Resolve casting exceptions
In Finance and Operations, X++ is completely intermediate-language (IL) based and therefore has a stricter runtime type behavior than the interpreted Dynamics AX2 012. This stricter runtime type behavior can generate exceptions in migrated Dynamics AX 2012 R3 metadata. It is likely you will encounter these exceptions during your migration. The casting exceptions can be raised in different runtime scenarios, such as down-casting, casting runtime to design time objects, and side-casting. In the section below, we will walk through an example where a form, CosJournalName, is generating controls at runtime, and has a type mismatch which causes a .NET exception because it is strongly typed.

Example: Side-casting exception

1. In Visual Studio, select and right-click Project Properties, and verify that USMF is the default company.

2. Add the display menu item CosJournalName to your project, and set the menu item as your Startup object.

3. Add the CosJournalName form to your project.

4. Add the cosDimCheckBoxController class to your project.

5. Rebuild your project.

6. Press Ctrl+F5 to run the form.

7. Note that you will get an exception, similar to the following, when running the form.

   You've received multiple errors.

   at Dynamics AX.Application.cosDimCheckBoxController.getBuildControl() in xppSource://Source/AxCosDimCheckBoxController.cs:line 5
   at Dynamics AX.Application.cosDimCheckBoxController.getDatasourceNum() in xppSource://Source/AxCosDimCheckBoxController.cs:line 10

   Unable to cast object of type 'Dynamics AX.Application.FormBuildCheckboxControl' to type 'Dynamics AX.Application.FormBuildStringControl'.

   Unable to cast object of type 'Dynamics AX.Application.FormBuildCheckboxControl' to type 'Dynamics AX.Application.FormBuildStringControl'.

   The menu item with name cosjournalname could not be opened.

   Close

8. Right-click the class, cosDimCheckBoxController, and then select View Code.

9. Set a breakpoint on the cosDimCheckBoxController::getBuildControl().


    - The breakpoint will be hit. This is where the casting error occurs. The reason for the casting error is because we are trying to return a control of type: FormBuildCheckboxControl and the object is expecting FormBuildStringControl.

11. Hover over the buildcontrol to see the type and notice the differences.
12. Press **F10** to hit the exception.

13. Stop debugging.

14. To fix the exception, change the method declaration from `FormBuildStringControl` to `FormBuildCheckBoxControl`.

```java
protected FormBuildStringControl getBuildControl()
protected FormBuildCheckBoxControl getBuildControl()
```

15. Rebuild the project, and press **Ctrl+F5**. The form should open successfully because the casting error is resolved.
Migrating context menus and mouse double-click code

Refer to these topics to migrate code Dynamics AX 2012 that deals with context menus and mouse double-click actions.

- Code migration - Context menu code
- Code migration - Mouse double-click logic
Configure the Azure DevOps mapping during code migration

This tutorial shows how to map your development box to the Azure DevOps project after the Lifecycle Services (LCS) code upgrade service has completed.

The LCS code upgrade service automatically checks your upgraded code into Azure DevOps. You will then need to map your development box to the upgrade folder/branch in your Azure DevOps project (The name of the upgrade folder/branch depends on the version you migrated to). Within your upgraded folder, you will find three folders:

- Export
- Metadata
- Projects

**Key concepts**

- **Export** is the project that contains the XML files after exporting from Microsoft Dynamics AX 2012. This project is your metadata in XML format before it is upgraded. This project is only relevant if you are upgrading from Dynamics AX 2012.
- **Metadata** is your upgraded code (metadata XML file).
- **Projects** are solutions that you can use during upgrade. One solution, CodeMergeSolution, is the solution that contains projects with the elements that have conflicts and need to be resolved. Another solution, UpgradedSolution, contains a collection of projects, one for each upgraded model.

**Map Azure DevOps to your development box**

1. In Visual Studio, connect to your account by going to Team Explorer > Select Team Projects > Servers > Add.
2. Enter the URL to your team project. Select Close.
3. Make sure the Azure DevOps account shows up. On the right, choose the project that you want to work on. Select Connect.
4. Now you need to map your workspace to the Azure DevOps folders. Go to the Source Code Explorer and do this mapping:
   a. Projects:
      - For Visual Studio 2015: C:\Users\<username>\Documents\Visual Studio 2015\Projects
      - For Visual Studio 2017 or newer: C:\Users\<username>\source\repos
   b. Metadata: C:\AOSService\PackagesLocalDirectory
      - On cloud VMs, this folder is located on the I\, J\ or K\ drive
      - On earlier versions, this folder is C:\packages
      - **Important:**
        - If you are migrating from Dynamics AX 2012 R3 or earlier, you will be mapping to the metadata folder under the Main branch.
        - If you are migrating between two product versions, you will be mapping to the metadata folder under one of the Releases branch.
After you have mapped these folders, you can synchronize the code to your local box. Right-click Metadata and select Get latest. Similarly synchronize the Projects folder. After synchronizing the metadata folder, refresh your models in Visual Studio from Finance and Operations > Model Management > Refresh Models.

You are now ready to open your projects, resolve conflicts, build, test, and complete your code migration.
This topic explains the split of the stack into three main models - the Application Platform, the Application Foundation, and the Application Suite.

Overview

Developing modular code is the driving force behind the model split. Splitting the stack into multiple models provides many benefits, including faster compile time and a greater distinction between partner's IP in production. There are three main models: the Application Platform, the Application Foundation, and the Application Suite.

Models

The Application Platform is the lowest model and contains the lowest level elements that interface with the kernel. The Application Object Server (AOS) can be started with only the Application Platform. The Application Foundation sits atop the Application Platform and contains framework functionalities that are shared by all applications. Finally, the Application Suite sits atop the Application Foundation and contains application specific elements. The Model Breakdown table in the appendix provides examples of components in each of these models. Each model is compiled into its own assembly with dependencies on lower layer model assemblies. The Application Platform does not depend on any other models. This implies a direct mapping of the model to an assembly.

Developing in the modular stack allows changes to be made in the Application Suite and compiled without touching the rest of the stack. Only models with new changes need to be compiled, greatly reducing compile time. More information can be found in the “Model Breakdown” section at the end of this article.
Customizing models

There are two methods for customizing: overlayering and extensions. Overlayering allows for changes to be made at multiple layers that alter, or overlay, elements in models at lower levels. Extensions allow for new elements to be added or code to be attached to element events or plug-in points. The type of customization used impacts how a model will be compiled and ultimately packaged. One or more models are compiled into an assembly. An assembly, its non-code metadata, and its compiled artifacts, form a package. A package is an independent deployable unit. A model that contains only extension customizations can be compiled into its own assembly and be deployed in its own package. A model that contains any overlayering must be compiled into the assembly based on the overlaid model.

Using extensions has several advantages, including:

- For application lifecycle management purposes, you need to manage only your extension artifacts.
- Building a customization does not require you to recompile the entire application.
- In the cloud, Microsoft can install, patch, upgrade, and change internal APIs without affecting your customizations.
- You can service your solutions independently without concerns about other customizations.

There are currently support code extensions, table extensions, form extensions, menu extensions, and enum extensions. The Extensions section in Customize through extension and overlayering and Customize model elements through extension provide a more detailed explanation on how to use extensions. Extensions should be used on supported elements wherever possible and are best applied when no change to existing Microsoft code is needed. A change to mask a method’s functionality requires overlayering to change the code itself. Overlayering should be in areas not covered by extensions and when the customization alters the base functionality. The illustration below summarizes differences between the two customization strategies.
Model breakdown

<table>
<thead>
<tr>
<th>Model</th>
<th>Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Platform</td>
<td>The Application Platform interfaces to kernel functionalities that are application logic agnostic. The AOS can be started with just this model.</td>
</tr>
<tr>
<td></td>
<td>• AIF base objects</td>
</tr>
<tr>
<td></td>
<td>• Batch</td>
</tr>
<tr>
<td></td>
<td>• Form base objects</td>
</tr>
<tr>
<td></td>
<td>• RunbaseSysOperations* base objects</td>
</tr>
<tr>
<td></td>
<td>• DictXX objects</td>
</tr>
<tr>
<td></td>
<td>• Appl, Info, Global, ClassFactory</td>
</tr>
<tr>
<td></td>
<td>• Data access objects</td>
</tr>
<tr>
<td></td>
<td>• Helper Classes</td>
</tr>
<tr>
<td></td>
<td>• ENUM/EDT/Macros/ConfigKey/LicenseCode</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application Foundation</th>
<th>Application foundation features:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Dimension framework</td>
</tr>
<tr>
<td></td>
<td>• Global Address Book</td>
</tr>
<tr>
<td></td>
<td>• Number Sequence</td>
</tr>
<tr>
<td></td>
<td>• OrgModel</td>
</tr>
<tr>
<td></td>
<td>Feature areas:</td>
</tr>
<tr>
<td></td>
<td>• Business Intelligence</td>
</tr>
<tr>
<td></td>
<td>• Reports</td>
</tr>
<tr>
<td></td>
<td>• Upgrade framework</td>
</tr>
<tr>
<td></td>
<td>• Security</td>
</tr>
<tr>
<td></td>
<td>• E-Signature</td>
</tr>
<tr>
<td></td>
<td>• Data Import/Export</td>
</tr>
<tr>
<td></td>
<td>• Workflow</td>
</tr>
<tr>
<td></td>
<td>SYS objects:</td>
</tr>
<tr>
<td></td>
<td>• Best Practices</td>
</tr>
<tr>
<td></td>
<td>• Checklist</td>
</tr>
<tr>
<td></td>
<td>• Policy</td>
</tr>
<tr>
<td></td>
<td>• RecordTemplate</td>
</tr>
<tr>
<td></td>
<td>Miscellaneous:</td>
</tr>
<tr>
<td></td>
<td>• Currency</td>
</tr>
<tr>
<td></td>
<td>• Unit of Measure</td>
</tr>
</tbody>
</table>
| Application Suite | Supply Chain Management  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Human Capital Management</td>
</tr>
<tr>
<td></td>
<td>Professional Services, etc.</td>
</tr>
</tbody>
</table>
This topic explains how delegate methods serve as a means for defining a contract between the delegate instance and the delegate handler.

Overview

Finance and Operations is split into several models, with each model in separate package. The principal 3 models are Application Platform, Application Foundation, and Application Suite. With the model split, a hierarchy has been created where a higher model can take dependencies and access elements in the models below, but not in models above. For example, in this setup, Application Suite has full access to its elements, Application Foundation’s elements, and Application Platform’s elements. Application Foundation can access its own elements and those of Application Platform. Finally, Application Platform can only access its own elements. To learn about models and packages, see Models and packages.

While the model split provides many benefits, it creates a problem when trying to access elements defined in higher models. Delegates are the recommended method for accessing elements in higher models from a lower model. Delegates are very similar to events in that when a delegate instance is invoked, a handler with compatible signature code is executed. This permits higher layer code, the handler, to be called by lower layer code, the delegate instance.

Create delegates and handlers

A delegate declaration must have three things:

- The delegate keyword
- Type void
- Empty method

Delegate methods serve as a means for defining a contract between the delegate instance and the delegate handler. A delegate takes no action itself. This is enforced by having a void type and having no code in the method.
delegate void applyDiscountDelegate(real _receiptTotal, EventHandlerResult _result)
{
}

Adding the `SubscribesTo` keyword to a method creates a static delegate handler. `SubscribesTo` requires the class name of the delegate, and the string name of the delegate method.

In order for a delegate to be properly handled, the delegate method declaration, the delegate instance, and the delegate handler must have the same method signature. For example, the delegate instance below takes two inputs, a real number and an EventHandlerResult, matching the delegate declaration and handler signatures above.

```csharp
[SubscribesTo(classStr(SimpleTax), delegateStr(SimpleTax, applyDiscountDelegate))]
public static void applyDiscountDelegateHandler(real _receiptTotal, EventHandlerResult _result)
{
    real discountedTotal = _receiptTotal * (1 - DiscountRate);
    _result.result(discountedTotal);
}
```

Due to the fact that delegates do not have a return value, an EventHandlerResult is passed as a parameter to provide access to the needed result value after the delegate has returned. This topic focuses on static delegate handlers using the `SubscribesTo`. The delegate functionality from Dynamics AX 2012 remains. *How to use X++ Delegates in Dynamics AX 2012* is a great blog post on MSDN by Microsoft developer Marcos Calderon on delegate concepts in Dynamics AX 2012. These concepts still apply.

**Example scenarios**

**Overlaying an existing delegate**

In many cases where delegates are needed, the code that was formerly overlayed has already been moved to a delegate handler by Microsoft. In these instances, Microsoft created delegates that can be leveraged and the code can be overlayed in a similar manner in the delegate handler. In this scenario, an Independent Software Vendor (ISV) is migrating code from Dynamics AX 2012 R3 where they have overlayed the `showSalesTax()` method in the `LogisticsEntityPostalAddressFormHandler` class. After migration, the CodeUpgrade project will contain the `LogisticsEntityPostalAddressFormHandler` with the *Your Solution, Microsoft AX 2012*, and *Microsoft AX* sections to resolve for the `showSalesTax()` method. The commented *Your Solution* section shows that the `showSalesTax()` method was overlayed by adding an additional table to approve showing sales tax from. This overlay is shown between the `<isv>` tags circled in red below.
When comparing this overlay with the code from Dynamics AX 2012, this is a simple change. The overlay has added an additional table to the switch statement.

```java
public boolean showSalesTax()
{
    boolean showTaxField;

    switch (this.getCallerRecord().TableId)
    {
    case tableNum(CustTable),
    tableNum(VendTable),
    tableNum(smmBusRelTable),
    tableNum(CompanyInfo),
    tableNum(InventSite),
    tableNum(InventLocation),
    //</.v>
    tableNum(MYVSTable),
    //</v>
    tableNum(HcmWorker):
        showTaxField = true;
        break;
        default:
        showTaxField = false;
    }
    return showTaxField;
}
```

However, the section for Finance and Operations does not appear to resemble either of the Dynamics AX 2012 code snippets.

```java
public boolean showSalesTax()
{
    boolean showTaxField = false;
    EventHandlerResult result = new EventHandlerResult();
    this.showSalesTax_delegte(this.getCallerRecord().TableId, result);
    if (result.result() != null)
    {
        showTaxField = result.result();
    }
    return showTaxField;
}
```
Upon deeper inspection, the code is calling a delegate method, `showSalesTax_delegate()`.

```csharp
this.showSalesTax_delegate(this.getCallerRecord().TableId, result);
```

The use of a delegate implies that code has been moved to another location. The `showSalesTax_delegate()` has been declared in the Application Foundation and handled in the Application Suite. To view the code that has been moved, find the delegate handler. The Finding Delegates and Handlers section contains methods to locate delegates and handlers. After finding the delegate handler method in the Application Suite, we see the code that has been moved from the `showSalesTax()` method. The same overlayered changes applied in Dynamics AX 2012 can be applied in the delegate handler.

```csharp
[SubscribesTo(classStr(LogisticsEntityPostalAddressFormHandler),
            delegateStr(LogisticsEntityPostalAddressFormHandler, showSalesTax_delegate))]
public static void onShowSalesTax_delegate(TableId _callerTableId, EventHandlerResult _res)
{
    switch (_callerTableId)
    {
    case tableNum(CustTable):
        break;
    case tableNum(VendTable):
        break;
    case tableNum(smmBusRelTable):
        break;
    case tableNum(CompanyInfo):
        break;
    case tableNum(InventSite):
        break;
    case tableNum(InventLocation):
        break;
    case tableNum(MarkWorker):
        _res.result(true);
        break;
    default :
        _res.result(false);
    }
}
```

After adding the new table to the switch statement in the delegate handler, the code will function as it did in Dynamics AX 2012.

```csharp
[SubscribesTo(classStr(LogisticsEntityPostalAddressFormHandler),
            delegateStr(LogisticsEntityPostalAddressFormHandler, showSalesTax_delegate))]
public static void onShowSalesTax_delegate(TableId _callerTableId, EventHandlerResult _res)
{
    switch (_callerTableId)
    {
    case tableNum(CustTable):
        break;
    case tableNum(VendTable):
        break;
    case tableNum(smmBusRelTable):
        break;
    case tableNum(CompanyInfo):
        break;
    case tableNum(InventSite):
        break;
    case tableNum(InventLocation):
        break;
    case tableNum(M茯Table):
        break;
    case tableNum(MarkWorker):
        _res.result(true);
        break;
    default :
        _res.result(false);
    }
}
```

Adding a new delegate

In this scenario, we will modify an existing tax calculation method that resides in the Application Foundation to account for discounts created in the Application Suite. The following class in the Foundation layer calculates the tax based on the gross total.
delegate void applyDiscountDelegate(real _receiptTotal, EventHandlerResult _result)
{
}

In the Application Suite, we have introduced the notion of discounts by adding a ProductDiscount class that contains the current discount.

public class ProductDiscount
{
    public static real DiscountRate;
}

The TaxCalculator class, in the lower Foundation layer, does not have access to the DiscountRate in the Suite layer and must use a delegate to update receipt total to use in the tax calculation. In the SimpleTax class, we create a delegate method, applyDiscountDelegate, with the state information that is needed by the handler in the signature. A delegate method is always empty because its only purpose is to define the contract between the delegate instance and the handler.

delegate void applyDiscountDelegate(real _receiptTotal, EventHandlerResult _result)
{
}

NOTE
The signature for the delegate declaration, the delegate instance, and the delegate handler must match. We now have to create an instance of the delegate at the point in the code where we would like the delegate handler to be run. The changes in between the <isv> tags represent the added code.

public real calculateTotalTax()
{
    // calculates tax on gross total.
    real totalTax;
    //<isv>
    EventHandlerResult result;
    this.applyDiscountDelegate(this.ReceiptTotal, result);
    this.ReceiptTotal = result.result();
    //</isv>
    totalTax = this.ReceiptTotal * this.TaxRate;

    return totalTax;
}

With the delegate in place, we now add a handler method in the Application Suite layer that has access to the discount information.
Using the SubscribesTo keyword, we tie the applyDiscountDelegate Handler method as a handler to the applyDiscountDelegate delegate.

NOTE
There can be more than one handler per delegate. There is not a defined order in the processing of handler methods. If order is important, delegate handler pairs should be chained together. With the final classes below, when the calculateTotalTax() method is run, the applyDiscountDelegate is fired and handled, updating the receiptTotal to provide an accurate tax calculation.

Full Code

SimpleTax class in the Application Foundation Layer

```java
public class SimpleTax {

    public real ReceiptTotal;
    public real TaxRate;

    public real calculateTotalTax() {
        // calculates tax on gross total.
        real totalTax;

        //\<i5>\</i5>
        EventHandlerResult result;
        this.applyDiscountDelegate(this.ReceiptTotal, result);
        this.ReceiptTotal = result.result();
        //\</i5>

        totalTax = this.ReceiptTotal * this.TaxRate;

        return totalTax;
    }

    //\<i5>\</i5>
    delegate void applyDiscountDelegate(real _receiptTotal, EventHandlerResult _result)
    {
    } //\</i5>
}
```

ProductDiscount class in the Application Suite layer

```java
//\<i5>\</i5>
public class ProductDiscount {
{
    public static real DiscountRate;

    //\<i5>\</i5>
    [SubscribesTo(classStr(SimpleTax), delegateStr(SimpleTax, applyDiscountDelegate))] public static void applyDiscountDelegateHandler(real _receiptTotal, EventHandlerResult _result) {
        real discountedTotal = _receiptTotal * (1-DiscountRate);
        _result.result(discountedTotal);
    }
    //\</i5>

Find delegates and handlers
There are three key ways to find delegates and handlers:

- Metadata search
- Class references
- SubscribesTo references

The Metadata search tool, described on the Metadata search in Visual Studio page, is the best way to find either delegates or their handlers. In Visual Studio, go to Dynamics 365 > Metadata Search to open the metadata search tool.

In the search field, type "code:<delegate name>" which will restrict the search to code and find any use of the delegate name, returning both the delegate and handler. Metadata search will search the entire code base and may take some time to complete, but will return any use of the search term in code.

Methods two and three can be used in parallel to the metadata search. The class where a delegate is defined can also serve as a means to narrow down the search for a delegate or handler. The SubscribesTo keyword requires the class name where the delegate was defined. Visual Studio's find references (right-click the class name > find references) will return a list of files that reference the class. This list will include both the class definition where the delegate is declared and the handler referencing the class. Finding class references is not a perfect method and will require some manual searching through class references. However, it produces a smaller subset of files and can be faster than a metadata search.
Similar to finding class references, finding all references can be done on the SubscribesTo keyword. The resulting list will include all static delegate handlers. Manually going through this list provides another means for finding static delegate handlers. This will not return dynamically declared delegate handlers that do not use the SubscribesTo keyword.
This topic explains the process of updating or upgrading to the latest release of Finance and Operations. It describes the overall process and supported scenarios, but it doesn’t provide detailed instructions for every step of the process.

For information about the contents of each release of Finance and Operations, see What's new or changed in Finance and Operations home page.

For information about One Version service updates, see the One Version service updates overview.

NOTE
For those looking to upgrade to Finance and Operations from Microsoft Dynamics AX 2012, please see Upgrade from AX 2012 to Finance and Operations.

Definitions

- **Upgrade** – The process of moving from one official release of Finance and Operations to the next release, for source environments prior to version 8.0. Some examples are the move from 7.1 to 7.3, or from 7.3 to 10.0.1. The process involves setup of a free sandbox environment, code upgrade, and data upgrade.

- **Update** – The process of applying a binary package to an environment to move it from one official release of Finance and Operations to the next release, for source environments starting with version 8.0. This process has lower downtime requirements and doesn’t involve data upgrade. For more information, see the Rebuild and update section later in this topic.

Paths to One Version

There are three primary paths to get to the latest version of Finance and Operations. Each path is referenced below with a link to detailed steps.

**Self-service upgrade**

*Applicable starting version: Microsoft Dynamics 365 for Finance and Operations 7.0 (RTW), 7.1 (1611), 7.2 (July 2017), 7.3.*

*Scope: Complex*

This path involves code refactoring to Extensions, and Data Upgrade in a DevTest, Sandbox, and eventually a Production environment.
Self-service upgrade to the latest version.

**Rebuild and update**

*Applicable starting version: Microsoft Dynamics 365 for Finance and Operations 8.0*

*Scope: Moderate*

This path involves removing Microsoft X++ hotfixes and creating a merged update package.

*Update environments from version 8.0 to 10.0.X.*

**Automatic update**

*Applicable starting version: Finance and Operations 8.1.0+*

*Scope: Simple*

This path involves configuring your project for continuous updates.

*Configure service updates through Lifecycle Services (LCS).*
IMPORTANT

The process that is described here is now deprecated for data upgrade between older versions of Finance and Operations apps and the latest version. For more information about Dynamic AX 2012 upgrades, see Upgrade from AX 2012 to Finance and Operations.

This topic applies to the following starting versions:

- Microsoft Dynamics 365 for Operations version 1611 (November 2016) (also known as version 7.1)
- Microsoft Dynamics 365 for Finance and Operations, Enterprise edition (July 2017) (also known as version 7.2)
- Microsoft Dynamics 365 for Finance and Operations, Enterprise edition 7.3

In this tutorial, you will learn how to perform these tasks:

- Understand which version to select.
- Refactor your customizations as extensions.
- Run the data upgrade in a development environment.
- Do a self-service upgrade in a sandbox user acceptance testing (UAT) environment.
- Do a self-service upgrade in a production environment.

Understand which version to select for upgrade

To align the self-service upgrade process to support continuous updates, each new release will cause the oldest release version to be discontinued.

For example, you have application version 7.3 with Platform update (PU) 23. Currently, the supported upgrade versions are 8.1.3 with PU 23, 10.0.0 with PU 24, and 10.0.1 with PU 25. When the next release, 10.0.2 with PU 26, is made generally available, it will be added to the available upgrade options, and 8.1.3 with PU 23 will be removed.

Because of this continuous process of adding a new version and removing the oldest version, we recommend that customers upgrade to the latest version that is available. In that way, you have two months in which you can upgrade your sandbox environment and then later upgrade your production environment to the same version.

If you choose to upgrade your sandbox environment to version 8.1.3 with PU 23 and Microsoft then releases version 10.0.2 with PU 26, so that version 8.1.3 with PU 23 is removed as an upgrade option, you will be blocked from upgrading your production environment. In this case, you must start over in the sandbox environment and upgrade to a newer supported version.

Targeted release schedule

NOTE

The dates in this table are subject to change. Upgrades are available a minimum of two weeks and a maximum of six weeks after the date of general availability (GA) for new customers. In addition, support for upgrades that have a target version of application 7.3 ends in March 2020.
<table>
<thead>
<tr>
<th>SELECTABLE VERSIONS</th>
<th>GA OF THE LATEST VERSION FOR NEW CUSTOMERS</th>
<th>GA OF THE LATEST VERSION FOR UPGRADE</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.3 with PU 23 – PU 25  8.1.3 with PU 23 – 10.0.1 with PU 25</td>
<td>Week of April 8, 2019</td>
<td>Week of April 29, 2019</td>
</tr>
<tr>
<td>7.3 with PU 24 – PU 26  10.0.0 with PU 24 – 10.0.2 with PU 26</td>
<td>Week of May 13, 2019</td>
<td>Week of May 27, 2019</td>
</tr>
<tr>
<td>7.3 with PU 25 – PU 27  10.0.1 with PU 25 – 10.0.3 with PU 27</td>
<td>Week of June 10, 2019</td>
<td>Week of June 24, 2019</td>
</tr>
<tr>
<td>7.3 with PU 26 – PU 28  10.0.2 with PU 26 – 10.0.4 with PU 28</td>
<td>Week of July 8, 2019</td>
<td>Week of July 29, 2019</td>
</tr>
<tr>
<td>7.3 with PU 27 – PU 29  10.0.3 with PU 27 – 10.0.5 with PU 29</td>
<td>Week of September 17, 2019</td>
<td>Week of September 30, 2019</td>
</tr>
<tr>
<td>7.3 with PU 28 – PU 30  10.0.4 with PU 28 – 10.0.6 with PU 30</td>
<td>Week of October 11, 2019</td>
<td>Week of October 28, 2019</td>
</tr>
<tr>
<td>7.3 with PU 29 – PU 31  10.0.5 with PU 29 – 10.0.7 with PU 31</td>
<td>Week of November 29, 2019</td>
<td>Week of December 30, 2019</td>
</tr>
<tr>
<td>7.3 with PU 30 – PU 32  10.0.6 with PU 30 – 10.0.8 with PU 32</td>
<td>Week of January 17, 2020</td>
<td>Week of February 17, 2020</td>
</tr>
<tr>
<td>7.3 with PU 31 – PU 33  10.0.7 with PU 31 – 10.0.9 with PU 33</td>
<td>Week of March 3, 2020</td>
<td>Week of March 30, 2020</td>
</tr>
<tr>
<td>10.0.8 with PU 32 – 10.0.10 with PU 34</td>
<td>Week of April 8, 2020</td>
<td>Week of April 27, 2020</td>
</tr>
</tbody>
</table>

**Refactor your customizations as extensions**

To prepare for upgrade, you must refactor any customizations that were overlays as extensions. We recommend that you deploy a new development environment on the latest version, create a new branch in version control, and follow the guidance in [Migrate from overlayering to extensions](#).

If you have no overlays and are already using extension for 100 percent of your customizations, we still recommend that you create a new branch for the upgrade effort in version control. If any Microsoft X++ hotfixes are installed, you must delete them from version control, because they aren't applicable to the latest version.

**Run the data upgrade in a development environment**

Run the data upgrade process on a copy of your source database. If your environment is already live in production, the source database is a copy of the production database. Otherwise, it's your most current database that is running the old version.

Run this process in the development environment that is running the release that you're upgrading to. This step is a validation process that is done by a developer. It helps the developer verify that the data upgrade can be successfully completed by using the specific set of customizations in the environment, without requiring any manual intervention.

To make a copy of your production database, follow the steps in [Export a copy of the standard user acceptance](#).
testing (UAT) database.

To run the data upgrade process, follow the steps in Upgrade data in development or demo environments.

**IMPORTANT**

- Data upgrade in a development environment is a required step. It helps reduce the risk of extended downtime and upgrade errors later, when you upgrade sandbox UAT and production environments.
- Several application hotfixes might be required before you can upgrade data. Before you redeploy your existing development environment, verify whether these hotfixes are required. Install the required hotfixes, and check them in to Microsoft Azure DevOps. This step can be completed only in the old version of your development environment. For a list of the hotfixes that are required in various situations, see Upgrade data in development or demo environments.

Upgrade your Tier 2+ Standard Acceptance Test sandbox environment

When you’ve completed the code upgrade and have been able to do an end-to-end data upgrade in your development environment without having to manipulate data in Microsoft SQL Server, you can begin the process in your sandbox environment.

**Prerequisite**

Before you begin your upgrade, we highly recommend that you make sure that your sandbox environment has the latest production data. If the data set is up to date, you can have more confidence that the upgrade will work in the production environment. To complete this step, use the Refresh for training purposes tutorial.

**IMPORTANT**

Changing Integrated Software Vendor (ISV) solutions, including changing the ISV license code/metadata, during upgrade is strictly not supported. If you are installing a new ISV solution or removing an existing ISV solution, you should do this before or after your upgrade. It cannot be performed during self-service upgrade.

**Begin the upgrade**

In your sandbox environment, on the Maintain menu, select Upgrade.

A dialog box appears, where you can select the latest combination of an application version and a platform update.
Prepare upgrade environment

Target environment name
App80Update3sb

Target version
Dynamics 365 for Finance and Op...

Once the staging environment is provisioned you will have 5 days to complete the upgrade and 5 additional days for final validations. If the upgrade has not been signed off by the end of this period the staging environment will be deleted.

**IMPORTANT**
If you receive an error that states that preparation failed, see the Known issues section later in this topic.

**Preparation**

The environment details page is refreshed, and options for two sandbox environments now appear in the upper-right corner. By selecting the options, you can switch between your old sandbox environment and your new upgrade-in-progress sandbox environment.

The preparation stage can take eight hours or longer, because it resembles a full environment deployment. The upgrade-in-progress environment is connected to an empty Azure SQL database to speed up deployment, and it runs on the newer version that you selected to deploy.

During this time, your original sandbox environment is left untouched. There is no downtime impact at this stage.

**IMPORTANT**
If you receive an error that states that staging deployment failed, the Microsoft Dynamics Service Engineering (DSE) team will be notified and will proactively resolve the issue for you. This issue can occur if Azure doesn't have the required resources available in your region. Microsoft DSE will work with the Azure engineers to allocate more resources. When staging deployment is successfully completed, you will receive an email.

**Package application**

After staging deployment is completed, go back to the environment details page, and switch to the Upgrade in progress view. In this view, you will now see an Upgrade menu.
The **Upgrade** menu includes an **Apply updates** option. You can select this option to apply your software deployable packages to the new environment. These packages include any binary packages, whether they are from an independent software vendor (ISV) solution, your own customization packages, or platform binary update packages.

**We highly recommend** that you apply the latest platform update as your first step. If you're upgrading to version 8.1, we recommend that you get the latest binary update package, such as 8.1.3. This package will also include the latest platform update. In this way, you help guarantee that you have the latest hotfixes that are available and help reduce errors later in the process.

When you apply a new package to the environment, the process is the same as the process for regular environment servicing. When package application is completed, you must use the **Sign Off** button for that package before you can move on or apply another package.

If package deployment fails, you can use the **Rollback** button to reverse it. Note that this button is **not** the same as the **Rollback** option on the **Upgrade** menu.

**Critical hotfixes**

As use of the self-service upgrade process has increased, Microsoft has found that several hotfixes are critical to success for various target versions. For example, if you're upgrading to version 7.3, a list of Microsoft Knowledge Base (KB) articles that have consistently resolved issues with data upgrade, Retail components, or performance will appear.

The goal is that this list should be empty before you begin the **Data Upgrade** step of the process. The hotfixes in these KB articles must be installed in your upgrade-in-progress environment.

**Data upgrade and environment swap**

After all packages are applied to your upgrade-in-progress sandbox environment, and you've signed off on
IMPORTANT
This stage begins the downtime for your original sandbox environment.

On the Upgrade menu, select Data upgrade.

Your original sandbox environment is turned off, and the database connection is swapped so that your new environment is connected to the original database. This process can take up to one hour.

Next, the data upgrade package for your target version is automatically applied. The time that is required to apply the data upgrade package varies, depending on the size of your database.

If the data upgrade fails, you must select Rollback on the Upgrade menu to restore your database to the point that it was at before the data upgrade began. Before you do a rollback, we highly recommend that you download the logs to determine the root cause of the failure. In this way, you can help guarantee that your next data upgrade execution will go more smoothly.

Upgrade days remaining
Because the self-service upgrade process provides a parallel environment at no additional cost to you, there is a time limit on how long this environment can be used. Currently, this time limit is set to 10 calendar days and begins when you select Upgrade on the Maintain menu to start the process.

What happens when the time limit expires?
There are three possible outcomes when the timer reaches 0 (zero):

- If you haven’t yet started the Data Upgrade step, the new environment is queued for deletion. In this scenario, the upgrade-in-progress environment was provisioned, and customizations and packages were optionally applied. However, no data was upgraded, and the original environment never incurred downtime.
- If you ran the Data Upgrade step but then later performed a rollback, the new environment is queued for deletion. In this scenario, the old environment is the primary environment, because the data upgrade was
If you’ve run the Data Upgrade step but haven’t yet committed the upgrade, no actions are performed, and no environments are deleted. You can remain in this state until you commit or do a rollback. If you decide to do a rollback, and the timer is at 0 (zero), the new environment will be deleted.

**IMPORTANT**

Rollback is only available, at maximum, for 30 calendar days. This is due to the nature of point-in-time restore. If you try to perform a rollback after 30 days have passed, you will be forced to commit the upgrade, delete the environment, and redeploy on the previous version.

The original environment is queued for deletion only after you commit the upgrade as a success.

**Commit or roll back**

After the data upgrade package is applied, you can review the environment, and your users can perform business validation activities. If this validation is successful, you can mark the whole upgrade as a success by selecting Commit on the Upgrade menu. You must commit the upgrade before you can move on to your production environment. After you commit the upgrade, the original environment is queued for deletion.

If the business validation fails, you can select Rollback on the Upgrade menu. This option will do a point-in-time restore of the database, swap the database connection back to your original sandbox environment, and bring your original sandbox environment back online. The sandbox environment will then be back in its previous state. Be aware, as stated above, that rollback is only possible for up to 30 calendar days.

**Post-upgrade actions**

After you’ve signed off on your upgrade, you must update aggregate measurements. Aggregate measurements must be updated after every major upgrade. To update them, go to System Administration > Setup > Entity Store, and then select Refresh.

**NOTE**

You can schedule this update to run by using batch processing.

**Upgrade production**

After you’ve committed the upgrade in the sandbox UAT environment, you’ve finished the upgrade process in the sandbox environment. You can now begin the same process in your production environment. The steps that you follow are the same.

If you encounter an issue that causes excessive downtime during your production upgrade, use the Report production outage process to alert Microsoft and get help.

**Upgrade additional environments**

You can upgrade additional sandbox environments in the same way. You also can deallocate and delete your
other sandbox environments, and then redeploy on the newer version. By using the Refresh database self-service action, you can copy in the upgraded database from another sandbox or production environment.

**Known issues**

Prepare operation could not start. Microsoft support has been notified. If the issue persists, please contact support with this ID.

This known issue involves environment certificates on the Microsoft Dynamics Lifecycle Services (LCS) back end. If it affects you, submit a support ticket, and include the activity ID from the error message. Microsoft will work to resolve the issue. Microsoft is compiling a list of affected environments and intends to proactively fix this issue in the future.

I want to cancel the upgrade and try again later.

To cancel an upgrade, you can select **Cancel Upgrade** on the **Maintain** menu. The **Maintain** menu is available in the **Old** view (for the original sandbox environment), not in the **Upgrade in progress** view (for the new sandbox environment).

Upgrade failed at step X: DVT script for service model: MRProcessService.

This DVT error is intermittent and can be resolved by using the **Resume** button for your data upgrade package. When you select **Resume**, the process resumes at the same step. Microsoft is trying to reliably reproduce this issue and intends to produce a fix in the future.

Application configuration sync failed. Call to TTSCOMMIT without first calling TTSBEGIN.

This TTSCOMMIT error is intermittent and can be resolved by using the **Resume** button for your data upgrade package. When you select **Resume**, the process resumes at the same step. (This issue is fixed in PU 21.)
This topic explains the steps required to update existing Finance and Operations 8.0 environments to 10.0.X application releases.

Background

Traditionally, moving to a newer application version has involved a rigorous upgrade that includes deployment of additional virtual machines, code upgrade, data upgrade, and scheduling several days in advance with the Microsoft Dynamics Service Engineering (DSE) team. You will notice that we are making the uptake of the latest version simpler, and this will continue to improve over time.

NOTE

We are supporting an update experience as compared to a full upgrade. This is possible because there are no Data Upgrade or Code Upgrade steps between the 8.0 and 10.0.X application schema. The target environments will be updated just like you would apply a Platform update.

The high-level process to update from version 8.0 to 10.0.X includes the following:

1. Deploy 10.0.X developer and build environments.
2. Branch in version control and remove any application hotfixes.
3. Recompile custom extensions and/or ISV solutions.
4. Produce a single software deployable package.
5. Merge a deployable package with the 10.0.X binary update package.
6. Deploy to target environments for validation.
7. Deploy to Production.

Deploy 10.0.X developer and build environments

Using Lifecycle Services, deploy at least one developer environment and a single, new build environment on application 10.0.X release.

On average this takes 3-4 hours and can be done simultaneously. For the build environment, Create a new agent pool and assign it to this environment on the Advanced options screen.

In Azure DevOps, visit your existing Build Definition and ensure that it is not using your new agent pool for 10.0.X. This will keep your new build agent from trying to compile older application code.

Apply the latest binary update to your build and development servers

In Lifecycle Services, go to the build server that you deployed in step 1. Using the Update tiles at the bottom of the Environment details page, grab the latest updates available and store it in your project's Asset Library using the Save package button. For example, this could be saving the 10.0 Platform update 24 package to the Asset Library.

Apply this same package back to the build server where you saved the package, as well as any of the new 10.0.X developer environments you have deployed.
Begin branch work for version control and remove any application hotfixes

While the new environments are deploying, begin the branching work for your update. Use the following branch structure in version control as an example. Branching design varies for each customer, so be careful to adjust your steps accordingly based on how your branches are set up.

Prepare using Visual Studio

On any other development machine (other than the new ones being deployed), open Visual Studio and visit the Source Control Explorer. You will create a new branch that will be isolated for the 10.0.X update.
Next, delete any Microsoft package folders in this branch. You can have packages, such as ApplicationSuite, checked in from applying hotfixes on 8.0 which need to be removed. When only your custom packages or ISVs remain, check these changes in to the branch.

IMPORTANT
It is critical that this is done before you map version control workspaces on your new development environments. This is to avoid the deletion of the Microsoft hotfixes to cascade to your working environment and delete untouched 10.0.X application code.

Recompile custom extensions and/or ISV solutions
Now you are ready to map this branch to a new development environment and compile your extensions and ISV solutions if they have provided you with source code. If your ISVs have only provided binary packages, you can check them in to source control, and the build environment will merge the binaries with your extension package to produce a single software deployable package. Additional information on this process can be found at Deployable packages from third parties. This will help later when you merge your package with the 10.0.X binary update.

NOTE
This step is necessary as the 10.0.X application code is not backward compatible with 8.0 from a binary level. In future application releases this step will be optional.

Produce a single software deployable package
After you have compiled in a developer environment and there are no errors to resolve, start a build in Azure DevOps using your new 10.0.X build environment agent that was setup earlier. When this is complete, a deployable package artifact will be attached to your build results. Download this package and upload it to the
Merge the deployable package with the 10.0.X binary update package

In your project's Asset Library, locate both your new 10.0.X software deployable package (your customization package that includes your ISVs) and the 10.0.X PU2X binary update package that was saved in Step 1 at the beginning of the topic. Highlight both packages and select Merge. This will combine the files into a merged update package. You can now apply this package to your various test environments.

**NOTE**

You can't move this merged package between different Lifecycle Services projects. The merge references other packages in your Asset library, and those packages won't be found in a different project.

Deploy to target environments for validation

Using the merged update package, deploy this to your various test environments. For more on how to do this, see Apply updates to cloud environments. This merged update package can be deployed to your Tier1/OneBox environments as well as Tier-2 sandboxes. At a minimum, you must deploy this to the sandbox Tier-2 environment that comes with your subscription. After you have finished with validation, mark the merged update package as a Release Candidate.

Deploy to Production

After you have marked the Release Candidate in your Asset Library, you can schedule the deployment to your Production environment. This will follow the same process for applying other software deployable packages.

Known issues

**GlobalUpdate script for service model: AOSService with error 'The specified module 'C:\Program Files\Microsoft Security Client\MpProvider'

This error is transient and can be ignored. To bypass, hit the Resume button from Lifecycle Services.

**Deploying the 10.0.X binary update to Developer environments causes ApplicationSuite compilation errors

The package can be applied to your 8.0 environments and it will update your source code. Compiling of your extension packages should not be impacted. If you had overlayering and have removed objects from the ApplicationSuite package, and try to recompile it, you may run into errors. Until this is resolved, please redeploy your developer environments on 10.0.X and sync in your source code from version control.

**Cannot find 10.0.X binary update package on the All Binary Updates tile on the My environment details page

It was originally communicated that the package would be found on the All Binary Updates tile. To prevent customers who want to simply get the latest binaries for release 8.0 from accidentally updating to release 10.0.X, we have moved the binary package to the Shared Asset Library. This topic has been updated to reflect this change.

**Deployment of my environment fails with error on duplicate objects

By default, in Visual Studio when an object is extended, it is created with a name of Object_EXTENSION1. This name could clash if Microsoft introduces new extensions of the same object. If this occurs, your deployment will fail with an error similar to the following:
Exception calling "CreateRuntimeProvider" with "1" argument(s): "Runtime metadata is invalid because the same metadata artifact has been defined in multiple assemblies. First 10 conflicting names: SystemAdministration.Extension1. See metadata events for complete list."

To prevent this from occurring, ensure that you compile your extensions on an 10.0.X developer machine. To resolve this issue, rename any of your extension objects with a vanity extension naming convention, such as SystemAdministration.Customer.

**Deployment on my environment fails with error on DVTs or ETWs**

There is a known issue where IIS/Application Pools are not fully restarted when the DVT or ETW step runs. The failure occurs because the DVTs are trying to connect to your environment's URL. To resolve this issue, click **Resume** on your deployment in LCS to retry the step. We are working to add a timer and automatic retry to resolve this issue.
This topic outlines the lifecycle and support policies for the Finance and Operations online service.

Modern Lifecycle Policy

The Finance and Operations online service is covered by the Modern Lifecycle Policy. The Modern Lifecycle Policy covers products and services that are serviced and supported continuously. For more information about this policy, see Modern Lifecycle Policy. Licensed customers must stay current with updates to the Finance and Operations online service in accordance with the following servicing and system requirements:

- Customers purchasing subscriptions of Finance and Operations and operating on the following application versions will experience continuous updates of the Platform and Financial Reporting. Microsoft will continually update these components with the option to postpone up to 3 consecutive service updates.
  - Dynamics 365 for Operations version 1611 (November 2016)
  - Dynamics 365 for Finance and Operations, Enterprise edition (July 2017)
  - Dynamics 365 for Finance and Operations, Enterprise edition 7.3
  - Dynamics 365 for Finance and Operations, version 8.0 (April 2018)
- Platform versions maintain backward compatibility with the application versions that are supported at the time of the platform release within the application support lifecycle. For more information about platform versions, see One Version service updates FAQ.
- Critical fixes and non-critical updates are handled in the following way:
  - **Critical fixes** – Critical fixes include security fixes and any fixes that are required to adhere to the availability service level agreement (SLA) that the service supports. Critical fixes will be made available in the latest platform update version and in the latest service update for customers operating on version 8.1. In addition, to help protect the customer and the online service, Microsoft might apply critical fixes directly to a customer’s environment. If a critical fix must be applied, Microsoft will notify the customer about the required downtime window (if there will be any downtime) and apply the fix to the applicable environment. The critical fix will update the system to the latest update version.
  - **Non-critical updates** – Customers operating on the following application releases must update to the most current Finance and Operations platform and financial reporter version to deploy non-critical updates.
    - Dynamics 365 for Operations version 1611 (November 2016)
    - Dynamics 365 for Finance and Operations, Enterprise edition (July 2017)
    - Dynamics 365 for Finance and Operations, Enterprise edition 7.3
    - Dynamics 365 for Finance and Operations, version 8.0 (April 2018)
  Customers operating on release 8.1 must update to the most current service update to deploy non-critical updates.
NOTE

Application and platform releases expire at the end of the month of their software lifecycle.

Microsoft will not provide any fixes to issues on versions that have reached end of service. Microsoft will also not investigate or troubleshoot any issue that you may encounter on an older version. If you encounter an issue on a version that has reached end of service, you will be required to update to the latest update and report the issue if it persists.

All environments will continue to be operated by Microsoft. All automatic processes around your environments, such as monitoring or self-healing, will also continue as is for supported versions.

Dates and versions for application and platform releases

Table 1: Continuous update releases

For information about the new features included in each release, click the links in the Version column.

<table>
<thead>
<tr>
<th>RELEASE</th>
<th>MAJOR RELEASE OR SERVICE UPDATE</th>
<th>VERSION</th>
<th>BUILD NUMBER</th>
<th>AVAILABILITY</th>
<th>END OF SERVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamics 365 for Finance and Operations</td>
<td>Major release</td>
<td>10.0</td>
<td>10.0.8</td>
<td>April 2019</td>
<td>Not applicable (continuously updated)*</td>
</tr>
<tr>
<td>Dynamics 365 for Finance and Operations</td>
<td>Major release</td>
<td>8.1</td>
<td>8.1.136</td>
<td>October 2018</td>
<td>Not applicable (continuously updated)*</td>
</tr>
</tbody>
</table>

* Indicates a major release is required to be updated through service updates. Service updates are cumulative in nature and may include updates for some or all of the following components: Platform, Application, Financial Reporting, Retail, and operating system updates. You will be required to have an update that’s no older than 3 service updates. The 8.1.x version series will be replaced by version 10.0, which is targeted for release in April 2019. For more information, see One Version service updates FAQ.

Table 2: Application releases

For information about the new features included in each release, select the links in the Version column.

<table>
<thead>
<tr>
<th>RELEASE</th>
<th>MAJOR OR MINOR RELEASE</th>
<th>VERSION</th>
<th>BUILD NUMBER</th>
<th>AVAILABILITY</th>
<th>END OF SERVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamics 365 for Finance and Operations</td>
<td>Major release</td>
<td>8.0</td>
<td>8.0.30</td>
<td>April 2018</td>
<td>April 30 2019</td>
</tr>
<tr>
<td>Dynamics 365 for Finance and Operations, Enterprise edition</td>
<td>Major release</td>
<td>7.3</td>
<td>7.3.11971.56116</td>
<td>December 2017</td>
<td>April 30 2019*</td>
</tr>
<tr>
<td>Dynamics 365 for Finance and Operations, Enterprise edition</td>
<td>Major release</td>
<td>July 2017</td>
<td>7.2.11792.56024</td>
<td>June 2017</td>
<td>April 30 2019</td>
</tr>
</tbody>
</table>
### Dynamics 365 for Operations

**Major release**

- **Version**: 1611
- **Build Number**: 7.1.1541.3036
- **Availability**: November 2016
- **End of Service**: April 30, 2019

### Dynamics AX

**Minor release**

- **Version**: 7.0.1
- **Build Number**: 7.0.1265.23014
- **Availability**: May 2016
- **End of Service**: June 2017

**Major release**

- **Version**: 7.0
- **Build Number**: 7.0.1265.3015
- **Availability**: February 2016
- **End of Service**: June 2017

---

*All customers must be on the latest version of Finance and Operations by April 2019. However, we are making an exception for customers who have unfulfilled extension requests that have been submitted to Microsoft. Those customers who submitted extensibility requests by January 1, 2019, will be supported on version 7.3 until their extensibility requests are fulfilled. Customers are expected to upgrade to the latest version within 90 days of the extensibility request being fulfilled. For more information, see One Version service updates FAQ.*

### Table 3: Platform releases

For information about the new features included in each release, select the links in the **Release** column.

<table>
<thead>
<tr>
<th>RELEASE</th>
<th>BUILD NUMBER</th>
<th>AVAILABILITY</th>
<th>EXPIRATION DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform update 31</td>
<td>7.0.5457</td>
<td>January 2020</td>
<td>N/A (Continuously updated)</td>
</tr>
<tr>
<td>Platform update 30</td>
<td>7.0.5407</td>
<td>November 2019</td>
<td>N/A (Continuously updated)</td>
</tr>
<tr>
<td>Platform update 29</td>
<td>7.0.5372</td>
<td>October 2019</td>
<td>N/A (Continuously updated)</td>
</tr>
<tr>
<td>Platform update 28</td>
<td>7.0.5314</td>
<td>July 2019</td>
<td>N/A (Continuously updated)</td>
</tr>
<tr>
<td>Platform update 27</td>
<td>7.0.5286</td>
<td>June 2019</td>
<td>N/A (Continuously updated)</td>
</tr>
<tr>
<td>Platform update 26</td>
<td>7.0.5257</td>
<td>May 2019</td>
<td>N/A (Continuously updated)</td>
</tr>
<tr>
<td>Platform update 25</td>
<td>7.0.5222</td>
<td>April 2019</td>
<td>N/A (Continuously updated)</td>
</tr>
<tr>
<td>Platform update 24</td>
<td>7.0.5179</td>
<td>March 2019</td>
<td>N/A (Continuously updated)</td>
</tr>
<tr>
<td>Platform update 23</td>
<td>7.0.5126</td>
<td>January 2019</td>
<td>N/A (Continuously updated)</td>
</tr>
<tr>
<td>Platform update 22</td>
<td>7.0.5095</td>
<td>December 2018</td>
<td>N/A (Continuously updated / Retired)</td>
</tr>
<tr>
<td>Platform update 21</td>
<td>7.0.5073</td>
<td>October 2018</td>
<td>N/A (Continuously updated / Retired)</td>
</tr>
<tr>
<td>Platform update 20**</td>
<td>7.0.5030</td>
<td>October 2018</td>
<td>N/A (Continuously updated / Retired)</td>
</tr>
<tr>
<td>Platform update 15*</td>
<td>7.0.4841</td>
<td>March 2018</td>
<td>N/A (Continuously updated / Retired)</td>
</tr>
<tr>
<td>Platform update 12</td>
<td>7.0.4709</td>
<td>November 2017</td>
<td>November 2018</td>
</tr>
<tr>
<td>Platform update 11</td>
<td>7.0.4679.35176</td>
<td>October 2017</td>
<td>October 2018</td>
</tr>
</tbody>
</table>
### Table 4: Application updates

The application updates listed below consist of a small subset of application enhancements released on top of Finance and Operations versions 8.0, 7.3, and 7.2 (July 2017). These updates do not affect the support lifecycle of the release—support is in-line with the policies for each release.

Note that application updates are not cumulative. The individual packages only contain the enhancements that were included in that specific release. However, if there is a dependency between two packages, then both packages will be included.

For information about the new features included in each update, click the links in the **Version** column.

<table>
<thead>
<tr>
<th>RELEASE</th>
<th>VERSION</th>
<th>BUILD NUMBER</th>
<th>AVAILABILITY</th>
</tr>
</thead>
</table>
| Dynamics 365 for Finance and Operations | 8.1.3: KB 4470000  
| Dynamics 365 for Finance and Operations | 8.1.2: KB 4470000  
Microsoft Dynamics 365 for Finance and Operations version 8.1.2 with Platform update 22* | 8.1.195      | December 2018 |
<table>
<thead>
<tr>
<th>RELEASE</th>
<th>VERSION</th>
<th>BUILD NUMBER</th>
<th>AVAILABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamics 365 for Finance and Operations</td>
<td>8.0.4: KB 4458992 Microsoft Dynamics 365 for Finance and Operations - Version 8.0.4 (Binary part)<em>, KB 4458993 Microsoft Dynamics 365 for Finance and Operations - Version 8.0.4 (X++ part)</em></td>
<td>8.0.35.15532</td>
<td>August 2018</td>
</tr>
<tr>
<td>Dynamics 365 for Finance and Operations</td>
<td>8.0.3: KB 4346176 Microsoft Dynamics 365 for Finance and Operations - Version 8.0.3 (Binary part)<em>, KB 4346172 Microsoft Dynamics 365 for Finance and Operations - Version 8.0.3 (X++ part)</em></td>
<td>8.0.35.15342</td>
<td>July 2018</td>
</tr>
<tr>
<td>Dynamics 365 for Finance and Operations</td>
<td>8.0.2: KB 4340414 Microsoft Dynamics 365 for Finance and Operations - Version 8.0.2 (Binary part)<em>, KB 4340143 Microsoft Dynamics 365 for Finance and Operations - Version 8.0.2 (X++ part)</em></td>
<td>8.0.35.15211</td>
<td>July 2018</td>
</tr>
<tr>
<td>Dynamics 365 for Finance and Operations</td>
<td>8.0.1: KB 4295107 Microsoft Dynamics 365 for Finance and Operations - Version 8.0.1 (Binary part)<em>, KB 4294515 Microsoft Dynamics 365 for Finance and Operations - Version 8.0.1 (X++ part)</em></td>
<td>8.0.30.15107</td>
<td>June 2018</td>
</tr>
<tr>
<td>Dynamics 365 for Finance and Operations, Enterprise edition</td>
<td>7.3.2: KB 4093261 Microsoft Dynamics 365 for Finance and Operations - Version 7.3.2 (Binary part)<em>, KB 4093262 Microsoft Dynamics 365 for Finance and Operations - Version 7.3.2 (X++ part)</em></td>
<td>7.3.11971.62687</td>
<td>March 2018</td>
</tr>
</tbody>
</table>
**Support matrix**

Platform updates are compatible with all application versions that are supported at the time of release.

**Table 5: Downloadable virtual hard drive (VHD) releases**

Use of the VHDs is subject to the [Software license terms](https://docs.microsoft.com/en-us/privacypolicies/microsoft-terms).

<table>
<thead>
<tr>
<th>RELEASE</th>
<th>VERSION</th>
<th>BUILD NUMBER</th>
<th>AVAILABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamics 365 for Finance and Operations, Enterprise edition</td>
<td>Application update 4: KB 4047325 Application Update 4 for Dynamics 365 for Finance and Operations (Binary part)<em>, KB 4047321 Application Update 4 for Dynamics 365 for Finance and Operations (X++ part)</em></td>
<td>7.2.11792.62509</td>
<td>October 2017</td>
</tr>
<tr>
<td>Dynamics 365 for Finance and Operations, Enterprise edition</td>
<td>Application update 1: KB 4035749 Application Update 1 for Dynamics 365 for Finance and Operations (Binary part)<em>, KB 4035751 Application Update 1 for Dynamics 365 for Finance and Operations (X++ part)</em></td>
<td>7.2.11792.62089</td>
<td>July 2017</td>
</tr>
</tbody>
</table>

* The link points to a Knowledge Base (KB) article. You must sign in to Lifecycle Services (LCS) to view the KB article.
<table>
<thead>
<tr>
<th>RELEASE</th>
<th>VHD NAME</th>
<th>VHD EXPIRATION DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform update 12 / Application release 7.2</td>
<td>FinandOps7.2PlatUpdate12.vhd</td>
<td>May 24, 2018</td>
</tr>
<tr>
<td>Platform update 12 / Application release 7.3</td>
<td>FinandOps7.3PlatUpdate12.vhd</td>
<td>June 05, 2018</td>
</tr>
<tr>
<td>Platform update 15 / Application release 7.3</td>
<td>FinandOps7.3withPlatUpdate15</td>
<td>December 08, 2018</td>
</tr>
</tbody>
</table>
This topic explains how to apply the latest platform release to your Finance and Operations environment.

Overview

In Finance and Operations, the platform consists of the following components:

- Binaries such as Application Object Server (AOS), the data management framework, the reporting and business intelligence (BI) framework, development tools, and analytics services.
- The following Application Object Tree (AOT) packages:
  - Application Platform
  - Application Foundation
  - Test Essentials

IMPORTANT

To move to the latest platform, your Finance and Operations implementation cannot have any customizations (overlayering) of any of the AOT packages that belong to the platform. This restriction was introduced in Platform update 3, so that seamless continuous updates can be made to the platform.

Overall flow

The following illustration shows the overall process for upgrading the platform to the latest update.

If you are already running on Platform update 4 or later, updating to the latest release is a simple servicing operation. After the platform update package is in your LCS asset library, follow the flow to apply an update from the LCS environment page. Select Apply updates under Maintain, then select the platform update package.
Learn how to get the latest platform package and apply it to an environment deployed through LCS in the next section.

Apply the latest platform update package

There are two ways to get the latest platform update package in LCS from your environment page.

- Click the **Platform binary updates** tile
- Click the **All Binary Updates** tile to see a list of combined package of application and platform binary updates. (As of Platform update 4, binary updates from LCS include an upgrade to the latest platform).

**NOTE**

Tiles on an environment’s page in LCS show only the updates that are applicable to your environment based on the current version and state of the environment.

Get the latest platform update package by clicking on one of the two tiles as mentioned above. After reviewing the fixes included in the platform, click **Save Package** to save the package to the project asset library.

From a process perspective, deploying a platform upgrade package resembles a binary hotfix deployable package.

- To apply a platform update package to your cloud development, build, demo, tier-2 sandbox, or production environment, update directly from LCS.

For more details, follow the instructions for applying a binary hotfix in **Apply updates to cloud environments**.
NOTE

Migrate files for Document management: After upgrading to Platform update 6 or later, an administrator needs to click the Migrate Files button on the Document management parameters page to finish the upgrade process. This will migrate any attachments stored in the database to blob storage. The migration will run as a batch process and could take a long time, depending on the number and size of the files being moved from the database into Azure blob storage. The attachments will continue to be available to users while the migration process is running, so there should be no noticeable effects from the migration. To check if the batch process is still running, look for the Migrate files stored in the database to blob storage process on the Batch jobs page.

Apply a platform update to environments that are not connected to LCS

This section describes how to apply a platform update package to a local development environment (one that is not connected to LCS).

How to get the platform update package

Platform update packages are released by Microsoft and can be imported from the Shared asset library in Microsoft Dynamics Lifecycle Services (LCS). The package name is prefixed with Dynamics 365 Unified Operations Platform Update. Use these steps to import the platform update package:

1. Go to your LCS project's Asset library.
2. On the Software deployable package tab, click Import to create a reference to the platform update package.
3. Select the desired platform update package.

NOTE

The package in the Shared Asset library may not correspond to the latest build (with hotfixes) of the desired platform release. To guarantee the latest build, use the LCS environment page as described earlier in this article.

Apply the platform update package to your development environment

NOTE

These instructions apply only to environments that cannot be updated directly from LCS.

Install the deployable package

1. Download the platform update package (AXPlatformUpdate.zip) to your virtual machine (VM).
2. Unzip the contents to a local directory.
3. Depending on the type of environment that you're upgrading, open the PlatformUpdatePackages.Config file under \AOS\Service\Scripts, and change the MetaPackage value.
   - If you're upgrading a development or demo environment that contains source code, change the MetaPackage value to dynamicsax-meta-platform-development.
   - If you're upgrading a runtime environment, such as a tier-2 sandbox environment or another
4. Follow the instructions for installing a deployable package. See Install deployable packages from the command line.

5. If you're working in a development environment, rebuild your application's code.

Example

```
AXUpdateInstaller.exe generate -runbookid="OneBoxDev" -topologyfile="DefaultTopologyData.xml" -servicemodelfile="DefaultServiceModelData.xml" -runbookfile="OneBoxDev-runbook.xml"
AXUpdateInstaller.exe import -runbookfile=OneBoxDev-runbook.xml
AXUpdateInstaller.exe execute -runbookid=OneBoxDev
```

**Install the Visual Studio development tools (Platform update 3 or earlier)**

**NOTE**

Skip this section if you are updating to Platform update 4 or later, development tools are automatically installed as part of installing the deployable package.

Update the Visual Studio development tools as described in Update the Visual Studio development tools.

**Regenerate form adaptor models**

Form adaptor models are required for test automation. Regenerate the platform form adaptor models, based on the newly updated platform models. Use the xppfagen.exe tool to generate the form adaptor models. This tool is located in the package's bin folder (typically, `j:\AosService\PackagesLocalDirectory\bin`). Here is a list of the platform form adaptor models:

- ApplicationPlatformFormAdaptor
- ApplicationFoundationFormAdaptor
- DirectoryFormAdaptor

The following examples show how to generate the form adaptor models.

```
xppfagen.exe -metadata=j:\AosService\PackagesLocalDirectory -model="ApplicationPlatformFormAdaptor" -xmllog="c:\temp\log1.xml"
xppfagen.exe -metadata=j:\AosService\PackagesLocalDirectory -model="ApplicationFoundationFormAdaptor" -xmllog="c:\temp\log2.xml"
xppfagen.exe -metadata=j:\AosService\PackagesLocalDirectory -model="DirectoryFormAdaptor" -xmllog="c:\temp\log3.xml"
```

**Install the Data Management service (Platform update 3 or earlier)**
After the deployable package is installed, follow these instructions to install the new Data Management service. Open a **Command Prompt** window as an administrator, and run the following commands from the `\DIXFService\Scripts` folder.

```bash
msiexec.exe /uninstall {5C74B12A-8583-4B4F-B5F5-8E526507A3E0} /passive /qn /quiet
```

If you're connected to Microsoft SQL Server Integration Services 2016 (13.0), run the following command.

```bash
msiexec /i "DIXF_Service_x64.msi" ISSQLSERVERVERSION="Bin\2012" SERVICEACCOUNT="NT AUTHORITY\NetworkService" /qb /lv DIXF_log.txt
```

If you're connected to an earlier release of Microsoft SQL Server Integration Services, run the following command.

```bash
msiexec /i "DIXF_Service_x64.msi" ISSQLSERVERVERSION="Bin" SERVICEACCOUNT="NT AUTHORITY\NetworkService" /qb /lv DIXF_log.txt
```

### Apply the platform update package on a build environment (Platform update 6 or earlier)

**NOTE**

Skip this section if you are updating to Platform update 7 or newer. This was a prerequisite step for build environments.

If the build machine has been used for one or more builds, you should restore the metadata packages folder from the metadata backup folder before you upgrade the VM to a newer platform update. You should then delete the metadata backup. These steps help ensure that the platform update will be applied on a clean environment. The next build process will then detect that no metadata backup exists and will automatically create a new one. This new metadata backup will include the updated platform. To determine whether a complete metadata backup exists, look for a `BackupComplete.txt` file in `I:\DynamicsBackup\Packages` (or `C:\DynamicsBackup\Packages` on a downloadable virtual hard disk [VHD]). If this file is present, a metadata backup exists, and the file will contain a timestamp that indicates when it was created. To restore the deployment's metadata packages folder from the metadata backup, open an elevated Windows PowerShell **Command Prompt** window, and run the following command. This command will run the same script that is used in the first step of the build process.

```powershell
if (Test-Path -Path "I:\DynamicsBackup\Packages\BackupComplete.txt") { C:\DynamicsSDK\PrepareForBuild.ps1 }
```

If a complete metadata backup doesn't exist, the command will create a new backup. This command will also stop the Finance and Operations deployment services and Internet Information Services (IIS) before it restores the files from the metadata backup to the deployment's metadata packages folder. You should see output that resembles the following example.
After the metadata backup has been restored, delete (or rename) the metadata backup folder (DynamicsBackup\Packages), so that it will no longer be found by the build process.

**Apply the platform update package**

After you’ve prepared your build environment for this update, apply the platform update package by using the same method that you use on other environments.

**Additional resources**

*Process for moving to the latest update of Finance and Operations*
Upgrade data in development or demo environments

11/24/2021 • 17 minutes to read • Edit Online

IMPORTANT
The process that is described here is now deprecated for data upgrade between older versions of Finance and Operations apps and the latest version. For more information about Dynamic AX 2012 upgrades, see Upgrade from AX 2012 to Finance and Operations.

This topic explains how to upgrade an older database to the latest Finance and Operations application release.

The topic provides instructions for upgrading your Finance and Operations database in a Tier 1 environment to the latest update. A Tier 1 environment is also known as a development, one-box, or demo environment.

In Tier 2 or higher environments, including Production, you will run through the self-service upgrade steps as outlined in Self-service upgrade to the latest version.

IMPORTANT
- You do not have to upgrade your database if you're updating to the latest platform of Finance and Operations. Platform updates are backward-compatible. This topic applies only to the process of upgrading between releases of Finance and Operations applications, such as an upgrade from Microsoft Dynamics 365 for Operations version 1611 (November 2016) to Finance and Operations 8.0.
- This process doesn't apply to the upgrade of document attachments that are stored in Microsoft Azure blob storage.
- All upgraded custom code has to be applied on the environment before running the data upgrade process.
- If you are on version 8.0 or later, there is no longer a data upgrade between application versions.

Before you begin

1. Back up your current database.

2. You must have a functional environment that is already successfully running the update.

3. In the source environment, you must install one of the following hotfixes, depending on the version that you're upgrading from. These hotfixes correct an issue in the SysSetupLog logic, so that the upgrade process can detect the version that you're upgrading from:

   - If you're upgrading from the November 2016 release (also known as 1611 or 7.1, build 7.1.1541.3036): KB 4023686, "Could not find source system version information' error when you upgrade to the latest Application Release."

   - If you're upgrading from the July 2017 release (also known as 7.2, build 7.2.11792.56024): No hotfix is required for this version.

   - After you install application hotfixes required in this step, run a full database synchronization. This step is especially important for golden database environments. A full database synchronization fills the SysSetupLog table, which is used when the database is upgraded. Don't run the database synchronization from Microsoft Visual Studio for this step, because the SysSetup interface won't be triggered. To trigger the SysSetup interface, run the following command from an Administrator
4. If you're upgrading to Dynamics 365 Finance version 10.0.9 or 10.0.10, install the quality updates in the destination environment before you run the data upgrade.

5. If you're upgrading a database that began as a standard demo data database, you must also run the following script. This step is required because the demo data contains bad records for some kernel X++ classes.

   ```sql
   delete from classidtable where id >= 0xf000 and id <= 0xffff
   ```

6. Make sure that all Commerce Data Exchange (CDX) jobs have been successfully run, and that there is no unsynchronized transactional data in the cloud version of the channel database.

**Select the correct data upgrade deployable package**

To obtain the latest data upgrade deployable package for a target environment that is running the latest update, download it from the Microsoft Dynamics Lifecycle Services (LCS) Shared asset library.

1. Sign in to LCS.

2. Select the Shared asset library tile.

3. In the Shared asset library, under Select asset type, select Software deployable package.

4. In the list of deployable package files, find the data upgrade package that corresponds to your upgrade.

   - If you're upgrading from AX 2012, the package name starts with AX2012DataUpgrade. Select the package that corresponds to the release you are upgrading to. For example, AX2012DataUpgrade-10-0.

   - If you're upgrading from a previous release to the latest 10.0.X release, the package name is DataUpgrade-10-0.

   - If you're upgrading from a previous release to a preview release, the package name contains PREVIEW. For example, DataUpgrade-10-0-2-PREVIEW.

5. Select the package that corresponds to the release that you are upgrading to.

**Upgrade the database**

1. Extract the data upgrade deployable package to C:\Temp or a location of your choice.

   **NOTE**

   Skip this step if this is a development environment that is connected to LCS and you are planning to execute the data upgrade process directly from LCS.

2. Import or restore a backup of the source database (the database that you will be upgrading) to the demo or development environment that is already running the latest update that you want to upgrade to. Leave the existing database in place, and name your new database imported_new.
3. Rename the original database by adding the suffix _orig. Rename the newly restored database so that it has the same name as the original database. In this way, the two databases switch places.

```sql
ALTER DATABASE <original Dynamics 365 database> MODIFY NAME = <original Dynamics 365 database>_ORIG
ALTER DATABASE imported_new MODIFY NAME = <original Dynamics 365 database>
```

4. Create a backup of the source database, in case you have to revert to it. This step is important because the following steps will modify the source database.

5. Execute the data upgrade package from the C:\Temp\DataUpgrade folder (the location that you extracted the deployable package to earlier). Executing a data upgrade package is similar to installing any software deployable package. For detailed instructions, see Install deployable packages from the command line. Start at the section titled Generate a runbook from the topology then execute the steps in the section Install a deployable package.

**NOTE**

If you are upgrading a database on a development environment, you can instead execute the data upgrade package directly from the LCS environment page, using the Maintain > Apply Updates servicing functionality. This does not require the user to be a local Administrator on the development VM. This is available as of the February release of LCS.

This will upgrade your Finance and Operations database, channel database, and reset the Financial reporting database.

---

**Re-enable SQL change tracking**

Run the following SQL against the upgraded database to make sure that change tracking is enabled at the database level. You must specify the name of your database in the alter database command.

```sql
ALTER DATABASE [<your AX database name>] SET CHANGE_TRACKING = ON (CHANGE_RETENTION = 6 DAYS, AUTO_CLEANUP = ON)
```

**Refresh the data entities list**

If you have upgraded to Platform update 14 or later, then you will need refresh the data entity list in the Data management workspace (Data management > Framework parameters > Entity settings > Refresh entity list) to ensure that the entity list is rebuilt on the latest platform and that the required metadata is available for data management operations.

**Troubleshoot upgrade script errors**

This section provides information that can help you troubleshoot various issues.

**Rerun the runbook after a data upgrade script failure**
A data upgrade deployable package enables the runbook to be rerun in a more granular manner than a typical deployable package. The data upgrade scripts begin to be run at Step 5 of the runbook. If you experience a failure during Step 5, view the output in the command window to learn which substep you reached. For example, if you reached substep 5.3, use the following command to rerun from that substep.

```bash
AXUpdateInstaller.exe execute -runbookid=upgrade -rerunstep=5.3
```

When you're debugging, you don't have to rerun the whole data upgrade piece and database synchronization. You can keep rerunning just the script that fails.

**View more details about a script error**

Upgrade scripts run in X++ by using a batch process that the runbook installer starts. In Application Explorer in Visual Studio, some classes that you can view are prefixed with `ReleaseUpdate`. If an upgrade script fails during the runbook process, you can learn more about the reason for the error by opening Microsoft SQL Server Management Studio and running the following code to query `ReleaseUpdateScriptsErrorLog`.

```sql
select * from RELEASEUPDATESCRIPTERRORLOG
```

You can add this code to a new runnable class in Visual Studio, and directly observe, debug, and rework its behavior.

**Skip failed scripts**

You can skip all scripts that have failed a specific number of times, and move to the next viable scripts. This functionality helps with the troubleshooting process. By design, the process is very manual, so that you're less likely to unintentionally skip scripts.

In the `ReleaseUpdateConfiguration` table, there is a new field that is named `ScriptRetryCount`. The value in this field controls how many times the runbook process will rerun scripts before it ignores them. When the runbook is run, the system updates the `ReleaseUpdateScriptsErrorLog.ErrorCount` field every time that a specific script fails. A new row is created for each script.

In the DataUpgrade package folder, under `\AosServices\Scripts\`, there is a script that is named `IgnoreBlockingScripts.ps1`. Run this script from an Administrator Windows PowerShell window to skip all scripts where `ScriptRetryCount = ErrorCount`. Then rerun the runbook step that failed, so that scripts will be ignored. The `ReleaseUpdateScriptsErrorLog.Ignored` field will also be set for each script that is skipped. Therefore, you can easily identify skipped scripts later.

**Monitor the duration of scripts that are run**

Every script that is successfully run records the number of minutes that it took in the `ReleaseUpdateScriptsLog.DurationMins` column. Therefore, you can easily identify the longest-running scripts when you're trying to tune the performance of the data upgrade process. Note that the duration is the amount of time that each script takes to run. However, because multiple scripts run in parallel, the sum of values in the `DurationMins` column will exceed the overall duration of the upgrade process.

**Known issues**

A duplicate key was found for the object that is named `dbo.RESOURCESETUP`.

When you upgrade a database, you might receive the following error message during the database upgrade:

```sql
A duplicate key was found for the object that is named dbo.RESOURCESETUP
```
A record can't be selected in Dimension hierarchy nodes (CAMDataDimensionHierarchyNode)

When you upgrade a database, you might receive the following error message during the database synchronization phase of the runbook process:

```
Cannot select a record in Dimension hierarchy nodes (CAMDataDimensionHierarchyNode). Dimension hierarchy: 0. The SQL database has issued an error. Object Server DynamicsAXBatchManagement: [Microsoft][ODBC Driver 13 for SQL Server][SQL Server]Invalid column name 'RELATIONTYPE'.
```

This issue is a known issue that will be resolved in a future release. The workaround is to create a missing field in several tables by running the following SQL script against the database from Management Studio.

```sql
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
DROP PROCEDURE IF EXISTS [dbo].PATCHRELATIONTYPE
GO

CREATE PROCEDURE [dbo].PATCHRELATIONTYPE
@TABLENAME SYSNAME
AS
BEGIN
    DECLARE @TABLEID  INT;
    DECLARE @FIELDID  INT;
    DECLARE @FIELDNAME  SYSNAME;
    DECLARE @SQLSTATEMENT NVARCHAR(1024);
    DECLARE @TOTALRECORDS INT;
    DECLARE @ParmDefinition NVARCHAR(150);

    SET NOCOUNT ON;
    SELECT @FIELDNAME = 'RELATIONTYPE';
    SELECT @FIELDID = 61453; --Hardcoded in code DBFIELD_DISCRIMINATOR
    IF OBJECT_ID(@TABLENAME, 'U') IS NULL
    BEGIN
        PRINT @TABLENAME + ' table does not exists. Please provide a base table';
        RETURN;
    END

    IF EXISTS(SELECT 1 FROM SYS.COLUMNS
               WHERE NAME = @FIELDNAME
               AND OBJECT_ID = OBJECT_ID(@TABLENAME))
    BEGIN
        PRINT @TABLENAME + ' table contains RelationType. No patching needed.';
        RETURN;
    END
    PRINT @TABLENAME + ' table does not contain RelationType.';
```
An index can’t be created on InventDistinctProduct

When you upgrade a database, you might receive the following error message during the database synchronization phase of the runbook process:

Cannot create index on InventDistinctProduct a duplicate key exists on column Product.

This issue is a known issue that will be resolved in a future release. The workaround is to delete all records in the InventDistinctProduct table and then resume the runbook from the current step. The records in the InventDistinctProduct table are disposable. They will be regenerated the first time that Finance and Operations is started, when an item is created, or when MRP is run. To delete all records in InventDistinctProduct, run the following query against the current database from Management Studio.

```
truncate table InventDistinctProduct
```

To resume the runbook from the current step, run the following command.

```
axupdateinstaller execute -runbookid=<your runbook name> -rerunstep=<the last step number>
```

For example, you can use this command.
axupdateinstaller execute -runbookid=dataupgrade -rerunstep=5.4

An exchange rate can't be found when demo data is upgraded
When you upgrade a demo database, you might receive the following error message when you deploy the data upgrade package:

An exchange rate cannot be found for exchange rate type Default between currencies INR and BRL on exchange date 12/1/2014.

Because you’re upgrading demo data, look in the TrvUnreconciledExpenseTransaction table, which is where the expense line is. Change the currency to USD. (Because the data is demo data, you don't have to be careful to preserve this expense line.)

update TrvUnreconciledExpenseTransaction
set transactioncurrencycode = 'USD'
where transactioncurrencycode = 'INR'

Alternatively, go to the original environment that the data came from (such as the old version), and add the missing exchange rate at General Ledger > Currencies > Currency exchange rates. You must add records for Indian rupee (INR) and Brazilian real (BRL) that cover 2014. Then bring that database into your new environment, and start the upgrade against that database.

The interpreter evaluation stack has grown during a call to the kernel
If you’ve enabled database logging on a kernel table such as UserInfom, you might receive the following error message:

Executing step: 5.1
prerq for data upgrade
prerq for data upgrade
Unhandled exception More Information: The interpreter evaluation stack has grown during a call to the kernel method xRecord::Delete (), height before call: 0, height after call: 3. Unhandled exception More Information: KernelInstance: Kernel is accessing deleted memory
The step failed

To resolve this issue, review the database log setup at System administration > Setup > Database log setup. Remove records for kernel tables as you require.

The batch process fails to start
The batch process can fail if the environment was left in maintenance mode after the configuration keys were changed. To resolve this issue, turn maintenance mode off, and then resume the runbook process.

The system fails to locate or generate a user GUID
You might receive the following error message:

...Unhandled exception More Information: System failed to locate or generate a user GUID...

To resolve this issue, rerun the runbook step that failed. You will then be able to continue.

Pre-sync or post-sync errors on ReleaseUpdateDB71_LedgerPeriodClose
You might receive one of the following error messages on the preSyncLedgerPeriodCloseTemplateTask, updateMenuItemTypeForCurrencyReval, or updateLedgerPeriodCloseTemplateTask method in the ReleaseUpdateDB71_LedgerPeriodClose class:
Cannot execute the required database operation. The SQL database has issued an error. Object Server DynamicsAXBatchManagement: [Microsoft][SQL Server Native Client 11.0][SQL Server]Invalid column name 'TEMPLATE'. INSERT INTO LedgerPeriodCloseTemplateTaskTmp (TEMPLATE, AREA, NAME, MENUITEM, MENUITEMTYPE, TARGETDAYSFROMPROJECTCOMPLETE, DUETIME, LEGALENTITYSELECTION, RECVERSION, PARTITION, RECID, CLOSINGROLE, LINENUM) SELECT T1.TEMPLATE, T1.AREA, T1.NAME, T1.MENUITEM, T1.MENUITEMTYPE, T1.TARGETDAYSFROMPROJECTCOMPLETE, T1.DUETIME, T1.LEGALENTITYSELECTION, T1.RECVERSION, T1.PARTITION, T1.RECID, T1.CLOSINGROLE, 0 FROM LedgerPeriodCloseTemplateTask T1 session 1013 (Admin) Microsoft.Dynamics.Ax.XppErrorException: Cannot execute the required database operation. The SQL database has issued an error.

Cannot execute the required database operation. The SQL database has issued an error. Object Server DynamicsAXBatchManagement: [Microsoft][SQL Server Native Client 11.0][SQL Server]Invalid column name 'MENUITEMTYPE'. UPDATE LedgerPeriodCloseTemplateTaskTmp SET MENUITEMTYPE = 0 WHERE MENUITEMTYPE = 2 AND MENUITEM = 'LedgerExchAdj' AND PARTITION = 5637144576 session 1013 (Admin) Microsoft.Dynamics.Ax.XppErrorException: Cannot execute the required database operation. The SQL database has issued an error.

Cannot execute the required database operation. The SQL database has issued an error. Object Server DynamicsAXBatchManagement: [Microsoft][SQL Server Native Client 11.0][SQL Server]Invalid column name 'TEMPLATE'. INSERT INTO LedgerPeriodCloseTemplateTask (TEMPLATE, AREA, NAME, MENUITEM, MENUITEMTYPE, TARGETDAYSFROMPROJECTCOMPLETE, DUETIME, LEGALENTITYSELECTION, RECVERSION, PARTITION, RECID, CLOSINGROLE, LINENUM) SELECT T1.TEMPLATE, T1.AREA, T1.NAME, T1.MENUITEM, T1.MENUITEMTYPE, T1.TARGETDAYSFROMPROJECTCOMPLETE, T1.DUETIME, T1.LEGALENTITYSELECTION, T1.RECVERSION, T1.PARTITION, T1.RECID, T1.CLOSINGROLE, T1.LINENUM FROM LedgerPeriodCloseTemplateTaskTmp T1 session 1013 (Admin)

To resolve this issue, use Management Studio to manually drop the LedgerPeriodCloseTemplateTaskTmp table from the database. Then rerun the runbook step. This issue will be fixed in a future hotfix.

**Table Sync Failed for Table: WarrantyGroupConfigurationItem**

If you're upgrading to Dynamics 365 Finance version 10.0.9 or 10.0.10, you might receive the following error message during data upgrade:

Table Sync Failed for Table: WarrantyGroupConfigurationItem

To resolve the issue, roll back the database upgrade, install the quality updates in the destination environment, and then rerun the data upgrade.

**KB number 3170386**

If KB number 3170386 isn't installed, you will receive the following error message:

GlobalUpdate script for service model: AOSService on machine …. Etc .... UpgradeServiceHelper::WaitForDataUpgradeToComplete(Object[])... The step failed

This error is caused by a failure in the pre-sync or the post-sync substep of the data upgrade. Follow these steps to determine which substep failed and the details of the failure.

**NOTE**

You can't rerun the failed runbook step until the pre-sync or post-sync substep has been manually completed, and the AutoDataUpgrade.config file has been updated to skip the substeps that have already been run.

1. In File Explorer, in the DataUpgradeAosServiceScripts folder, sort by descending order of the date
When files were last modified, and then look at the file at the top of the list to determine which substep failed.

- If the top file is named `dbUpgradePreSyncMonitor.error.log`, the pre-sync substep failed.
- If the top file is named `dbUpgradePostSyncMonitor.error.log`, the post-sync substep failed.

2. In Management Studio, run the following `SELECT` statement.

```
SELECT * FROM RELEASEUPDATELOG
```

The second-to-last record in the result set will have the errors and call stacks for all the failures.

**DMF errors**

If you receive either of the following Data Migration Framework (DMF) errors, download hotfix KB number 3170386, and then restart the upgrade process.

**DMF pre-sync error**

```
Batch error: initial.DAT.ReleaseUpdateDB70_DMF.updateIntegrationActivityExecutionMessageIdPreSync
(Batch:AOS-F01B9F0CCC8, 9, Info, Error, ):[1][3,Cannot execute the required database operation. The SQL
database has issued an error][3,Object Server DynamicsAXBatchManagement: ]][3,[Microsoft][SQL Server
Native Client 11.0][SQL Server]Incorrect syntax near 'GO'.][3,
```

**DMF post-sync error**

```
Batch error: initial.DAT.ReleaseUpdateDB70_DMF.updateIntegrationActivityExecutionMessageIdPostSync
(Batch:AOS-F01B9F0CCC8, 9, Info, Error, ):[1][3,Cannot execute the required database operation. The SQL
database has issued an error][3,Object Server DynamicsAXBatchManagement: ]][3,[Microsoft][SQL Server
Native Client 11.0][SQL Server]Incorrect syntax near 'GO'.][3,
```

**Encrypted fields in demo data**

After upgrade, values in encrypted fields in the database will be unreadable. However, new values that are entered in these fields after upgrade will be readable. This behavior occurs because of a technical limitation that is related to the certificate that is used for data encryption. The following table shows the fields that are affected.

<table>
<thead>
<tr>
<th>TABLE.FIELD</th>
<th>DATA EXISTS IN DEMO DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreditCardAccountSetup.SecureMerchantProperties</td>
<td>Yes</td>
</tr>
<tr>
<td>ExchangeRateProviderConfigurationDetails.Value</td>
<td>Yes</td>
</tr>
<tr>
<td>RetailChannelPaymentConnectorLine.SecureMerchantProperties</td>
<td>Yes</td>
</tr>
<tr>
<td>RetailConnDatabaseProfile.ConnectionString</td>
<td>Yes</td>
</tr>
<tr>
<td>RetailHardwareProfile.SecureMerchantProperties</td>
<td>Yes</td>
</tr>
<tr>
<td>RetailHardwareProfileMerchantInfoEntity.SecureMerchantProperties</td>
<td>Yes</td>
</tr>
<tr>
<td>FiscalEstablishment_BR.ConsumerEDocCsc</td>
<td>No</td>
</tr>
<tr>
<td>FiscalEstablishmentEntity.CSC</td>
<td>No</td>
</tr>
<tr>
<td>TABLE FIELD</td>
<td>DATA EXISTS IN DEMO DATA</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>FiscalEstablishmentStaging.CSC</td>
<td>No</td>
</tr>
<tr>
<td>HcmPersonIdentificationNumber.PersonIdentificationNumber</td>
<td>No</td>
</tr>
<tr>
<td>HcmWorkerActionHire.PersonIdentificationNumber</td>
<td>No</td>
</tr>
<tr>
<td>SysEmailSMTPPassword.Password</td>
<td>No</td>
</tr>
<tr>
<td>SysOAuthUserTokens.EncryptedAccessToken</td>
<td>No</td>
</tr>
<tr>
<td>SysOAuthUserTokens.EncryptedRefreshToken</td>
<td>No</td>
</tr>
</tbody>
</table>

**Additional resources**

Process for moving to the latest update of Finance and Operations
This topic explains how to update the development tools.

Use this tutorial to update your Visual Studio development tools with a new version. It explains how to uninstall your existing Visual Studio development tools and install the new extension. The new extension is in the form of an installable VSIX file. This file is a part of the binary hotfix available on the Dynamics Lifecycle Services (LCS) site. The VSIX file is located in the DevToolsService\Scripts folder of the binary hotfix package.

NOTE
You do not need to follow the instructions in this article if you are upgrading your Finance and Operations platform to Platform update 4 or newer. It is an automatic step that is part of the platform upgrade process.

Uninstall the existing Visual Studio extension

In order to install a new version of the development tools, you'll need to uninstall the existing version first. Verify the version of the development tools that you have installed. If you don't have it installed, you can skip this section.

Verify your current version of the Visual Studio extension
2. Select it and click OK.

Uninstall the extension
1. Open the Visual Studio Tools > Extensions and Updates dialog.
3. When the extension is uninstalled, exit Visual Studio.

Install a new version of the extension
1. Make sure Visual Studio is not running.
2. Double-click (or right-click and Open) the VSIX file of the new version.
3. Follow the installation instructions.
4. When installation is complete, you can start Visual Studio and start developing your application.
With the rollout of the One Version servicing plan, Microsoft is committed to backward compatibility from a binary and functional perspective. For detailed information about One Version, see One Version service updates FAQ. Even with backward compatibility as a priority, there are situations where development activities may result in required code changes. Some of those situations are described below.

- When Microsoft makes an enumeration extensible, it is considered a binary compatible change. The compiler checks for unsafe extensible enumeration operations that depend on the integer value of a non-extensible enumeration. Any partner code that contains unsafe extensible enumeration operations will have compiler errors when re-compiled and will need to be modified. For more information, see Add values to enums through extension.

- To avoid possible unresolved references when compiling, a partner model should reference the top-level modules and sub-modules. If this is not done, a Microsoft change that adds new resources in an unreferenced sub-module may cause an unresolved reference. To resolve the compilation error, add the sub-module as a reference.

- Some methods will be attributed as obsolete to signal that they will be fully deprecated in the future. Any compiler warning that is generated due to the calling or wrapping of an obsolete method should be investigated to ensure that the expected code path still exists. In some cases, Microsoft code will directly call the new method in place of the obsolete method. When this happens, the code built around the obsolete method will not execute when expected.
This topic explains how to apply supported updates to Dynamics 365 Finance + Operations (on-premises). All updates to on-premises environments are done through Microsoft Dynamics Lifecycle Services (LCS).

Search for and download updates

For more information about how to find the updates that you can apply to your on-premises environment, see Issue search in Lifecycle Services (LCS). For information about how to download updates from the tiles in the Updates section of the Environment details page in LCS, see Download updates from Lifecycle Services (LCS).

NOTE
When you are updating an on-premises environment, always select updates from the update tiles on the Environment details page. If you select updates from another location, the updates might not work.

Update an on-premises deployment

You can apply updates to an on-premises environment either during deployment or after the deployment is completed.

While an on-premises environment is being deployed, you can select to deploy a custom package in the Advanced settings. For more information about how to apply customizations or application X++ updates, see Develop and deploy custom models to on-premises environments.

To apply updates to an on-premises environment after it has been deployed, in LCS, on the Environment details page for the environment, under Maintain, select Apply updates.

NOTE
You can apply updates after deployment only on environments that have Platform update 12 for Finance and Operations or later. The environment must also have the latest version of the local agent available in LCS. For more information, see Update the local agent. If you're on a platform version that is older than Platform update 12, you can reconfigure an environment that is already deployed to update the customizations or update to the latest platform release. For more information about how to redeploy an environment, see Redeploy on-premises environments.

Apply application or binary updates through LCS

The following steps can be used to apply X++, All Binary, or Platform binary updates.

IMPORTANT
The application of updates requires downtime for your environment. Therefore, no business transactions can be performed in the environment during the update. When you complete the following steps, verify that the system isn't being used, and that an official downtime notice has been communicated to all system users.
Update a sandbox environment

1. In the LCS Asset library, upload the deployable package that was generated in the "Prerequisites" section of this topic to the Software deployable packages tab.
2. In LCS, open the on-premises implementation project, and then open the Environment details page of the environment to update.
3. Under Maintain, select Apply updates. A slider shows the updates that were uploaded to the Asset library. Note that only packages that are marked as Valid in the Asset library appear.

If you are on local agent version 2.1.0 and higher, complete the following steps.

1. Select the update, and then click Prepare. Clicking on Prepare will prepare your on-premises environment for servicing.

   **NOTE**
   During preparation, the environment state will be Deployed but the Deployment status field will show the progress of Preparation. Steps such formatting the package and downloading the package are executed during preparation. The environment is not directly touched during preparation and hence there is no downtime during the preparation phase. Users can continue to use the system during preparation.

2. After the preparation is complete, you will see Abort and Update Environment buttons. To start applying the update, click Update Environment. If preparation fails, see the "Resolve a failed update application" section later in this topic.
3. In the confirmation message, select Yes. The servicing operation has started on this environment. This is the start of the downtime on your environment.
4. The environment state is changed from Deployed to Deploying.
5. After the update is completed, the environment state is changed back to Deployed. If application of the
update fails, the environment state is changed to Failed. For information about what to do if package application fails, see the "Resolve a failed update application" section later in this topic.

6. Open the History and Environment details pages to view the operations that were performed on the environment. You can also view a record of major actions that were performed on the environment, such as deployments, servicing, and rollbacks.

If you are on local agent version lower than 2.1.0, complete the following steps.

1. Select the update, and then click Apply.

2. In the confirmation message, select Yes. The servicing operation has started on this environment. This is the start of the downtime on your environment.

3. Environment state changes from Deployed to Preparing.

   **NOTE**
   During preparation, steps such formatting the package and downloading the package are executed during preparation. The environment is not directly touched during preparation and hence there is no downtime during the preparation phase. Users can continue to use the system during preparation. However, we recommend that the downtime starts when the environment enters the Preparing state.

4. After preparation is complete, the environment state is changed from Preparing to Deploying.

5. After the update is completed, the environment state is changed back to Deployed. If application of the update fails, the environment state is changed to Failed. For information about what to do if package application fails, see the "Resolve a failed update application" section later in this topic.

6. Open the History and Environment details pages to view the operations that were performed on the environment. You can also view a record of major actions that were performed on the environment, such as deployments, servicing, and rollbacks.

**Update a production environment**
Before you update a production environment, you must successfully complete the package application update on a sandbox environment.

1. In the project for the sandbox environment that you applied the package to, open the Asset library, and then, on the Software deployable packages tab, select the package, and mark it as a Release candidate.

2. On the Environment details page, under Maintain, select Apply updates. In the dialog box, only packages that are marked as a Release candidate are shown.

3. Select the Release candidate package to be applied to the Production environment.

4. The rest of the Update flow is the same as that of a sandbox environment. Your update experience will differ based on the version of the local agent running on your environment. We recommend that you always run with the latest version.

**Resolve a failed update application**
When preparation fails, the environment state is Deployed. When the application of an update fails, the environment state is Failed. The first step is to determine why there is a failure. The location of the logs varies, depending on the stage where the failure occurred:

- **Preparation stage:** If the operation fails during the Preparation stage, the logs are uploaded to LCS. In the log files, select Download logs to download the log files. If the package has any merge issues, the error is included in the log file.

- **Deploying stage:** If the operation fails during the Deploying stage, the logs are located in the on-premises
environment. You must sign in to the environment, and then access the logs and event viewer.

For more information about how to use the troubleshooting logs, see Troubleshoot on-premises deployments.

After you review the logs and determine the cause of the failure, complete one of the following operations to restore the environment to a healthy state. No actions can be performed on an environment that is in a Failed state. The environment must first be restored to a healthy state.

- **Retry failed operation** – If update application fails, select Retry to recover from the failed operation.
- **Abort failed operation** – Because there is no change made to the on-premises environment, if the preparation fails, you have the option to cancel the operation. Select Abort to cancel the preparation.
- **Roll back the update** – To roll back the update that failed, select Rollback. Before you start the rollback, you must restore the database to the last known good state. When you select Rollback, the environment is restored to the last known good state. The environment state is then changed to Preparation, then to Deploying, and then to either Deployed or Failed.

  **NOTE**
  
  The Rollback button doesn't roll back the database. You're responsible for restoring the database to the last known backup that was made before update application. This step is critical to help guarantee that there is no data loss.

- **Refresh the state** – If update application fails during the Preparation stage, the failure is on the LCS side, and update application hasn’t yet started. Therefore, the on-premises environment is in a good state. To restore the LCS environment state to Deployed, on the project dashboard page, select Refresh.

- **Delete and redeploy an environment** – If the retry and rollback options don't work, you must delete and redeploy the environment. To delete the environment, on the project dashboard page, select Delete. You then see the option to configure the environment.

  **IMPORTANT**
  
  This option should not be used on a production environment. However, it can be used on a sandbox deployment to restore the environment to a healthy state.

Because this option requires that you do a fresh deployment of the environment, you lose any updates that were previously applied. Any customizations and binary updates must be reapplied to the environment.
At some point, you might have to redeploy your on-premises environment. This could be to apply a new platform update or because of changes or issues in your implementation. Before you delete the environment you are currently working with, you should save your configuration setting information to use when you redeploy. This topic describes how to save configuration settings and how to redeploy your environment.

Save your configuration

Before you delete the environment you plan to update, use the following steps to save your configuration.

1. In LCS, navigate to Project Settings > On-prem Connectors.
2. Select the connector to your environment, and then click Edit.
3. On the Edit connector tab, navigate to Configure Agent > Enter Configuration.
4. Copy the value of the Download Fileshare location in the Configuration Settings section. You will need this later.
5. Log in to the on-premises environment file share machine and copy the \agent\wp<environment name>\StandaloneSetup\config.json. You can use the configuration settings in this json file to redeploy your environment.

Configuration settings

The following tables provide information about configuration settings. Use the Configuration setting value from the .json file that you saved in the previous procedure.

### Active Directory Federation Services settings

<table>
<thead>
<tr>
<th>FIELD</th>
<th>CONFIGURATION SETTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>The email address of the user who will be the initial administrator (such as, <a href="mailto:adminuser@yourdomain.com">adminuser@yourdomain.com</a>)</td>
<td>components. (AOS).parameters.provisioning.adminPrincipalName.value</td>
</tr>
<tr>
<td>ADFS OpenID metadata endpoint for the Dynamics 365 Application group. (such as, https://[federation-service-name]/adfs/.well-known/openid-configuration)</td>
<td>components. (AOS).parameters.activeDirectory.adfsMetadata.value</td>
</tr>
<tr>
<td>ADFS OpenID Connect client ID for the AOS application group</td>
<td>components. (AOS).parameters.activeDirectory.adfsClientId.value</td>
</tr>
<tr>
<td>ADFS OpenID Connect client ID for the Financial Reporting application group</td>
<td>components. (FinancialReporting).parameters.aad.nativeClientAuthentication.clientId.value</td>
</tr>
</tbody>
</table>

### SQL database configuration

<table>
<thead>
<tr>
<th>FIELD</th>
<th>CONFIGURATION SETTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL SERVER</td>
<td>components.(AOS).parameters.database.dbServer.value</td>
</tr>
<tr>
<td>AX DATABASE</td>
<td>components.(AOS).parameters.database.dbName.value</td>
</tr>
</tbody>
</table>
**FINANCIAL REPORTING DATABASE**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>CONFIGURATION SETTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>FINANCIAL REPORTING DATABASE</td>
<td>components. (FinancialReporting).parameters.mrdb.dbName.value</td>
</tr>
</tbody>
</table>

**File share settings**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>CONFIGURATION SETTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>The file share path for the Microsoft Dynamics 365 instance. This share is used as the document store for files uploaded by users.</td>
<td>components.(AOS).parameters.storage.fileSharePath.value</td>
</tr>
<tr>
<td>The File share certificate thumbprint for the Microsoft Dynamics 365 instance.</td>
<td>components. (AOS).parameters.storage.sharedAccessThumbprint.value</td>
</tr>
</tbody>
</table>

**NOTE**

When you copy the file path configuration value from .json file to LCS UI, make sure to remove the extra backslashes. For example, configuration value `\\DC1\D365FFOStorage` from the .json file should be `\DC1\D365FFOStorage` in the LCS UI.

**SSRS configuration settings**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>CONFIGURATION SETTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>The IP Address of the SSRS instance.</td>
<td>components. (AOS).parameters.biReporting.persistentVirtualMachineIPAddressSSRS.value</td>
</tr>
<tr>
<td>The thumbprint used by the SSRS application to communicate with AX Service.</td>
<td>components. (ReportingServices).parameters.reportingClientCertificateThumbprint.value</td>
</tr>
</tbody>
</table>

**Configure service settings**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>CONFIGURATION SETTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>DYNAMICS 365 DNS INFORMATION - The DNS host name of the Microsoft Dynamics 365 instance, such as ax.d365ffo.onprem.contoso.com.</td>
<td>components. (AOS).parameters.infrastructure.hostName</td>
</tr>
<tr>
<td>AOS SERVICE PRINCIPAL USER SETTINGS - The domain user account to run the AX service, such as yourdomain\axserviceuser.</td>
<td>components. (AOS).parameters.infrastructure.principalUserAccountName *</td>
</tr>
<tr>
<td>MR SERVICE PRINCIPAL USER SETTINGS - The group managed service account (gMSA) to run the MR application service, such as yourdomain\Svc-FRAS$.</td>
<td>components. (FinancialReporting).parameters.ApplicationServicePrincipalUserServicePrincipalUserAccountName.value *</td>
</tr>
<tr>
<td>The group managed service account (gMSA) to run the MR process service, such as yourdomain\Svc-FRPS$.</td>
<td>components. (FinancialReporting).parameters.ProcessServicePrincipalUserServicePrincipalUserAccountName.value *</td>
</tr>
</tbody>
</table>
The group managed service account (gMSA) to run the MR click-once service, such as yourdomain\Svc-FRCO$.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>CONFIGURATION SETTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>components. (FinancialReporting).parameters.ClickOnceServicePrincipalUse r.accountName.value *</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE**
Remove the extra backslash from the Principal username configuration value in the .json file before entering in the LCS UI. For example, contoso\AXServiceUser should be entered as contoso\AXServiceUser in LCS.

**Application certificate settings**

<table>
<thead>
<tr>
<th>FIELD</th>
<th>CONFIGURATION SETTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>The thumbprint of the Data Encryption certificate.</td>
<td>components. (AOS).parameters.database.dataEncryptionCertificateThumbprint.value</td>
</tr>
<tr>
<td>The thumbprint of the Data Signing certificate.</td>
<td>components. (AOS).parameters.database.dataSigningCertificateThumbprint.value</td>
</tr>
<tr>
<td>The thumbprint of the Session Authentication certificate.</td>
<td>components. (FinancialReporting).parameters.sessionAuthenticationCertificateThumbprint.value</td>
</tr>
<tr>
<td>The thumbprint of the SSL certificate used for WCF/SOAP support.</td>
<td>components. (AOS).parameters.infrastructure.sslCertificateThumbprint.value</td>
</tr>
<tr>
<td>The thumbprint used by the Management Reporter to communicate with AX service.</td>
<td>components. (FinancialReporting).parameters.tokenSpec.certThumbprint.value</td>
</tr>
</tbody>
</table>

**Redeploy your environment**

The following instructions provide information about how to update or redeploy your environment with a new platform or topology.

1. In LCS, navigate to the **Environments** blade in your on-premises project.

2. Click **Delete** to delete your environment.

   **NOTE**
   Deleting the environment will not delete the database, infrastructure or Local agent. Only the Service Fabric applications are deleted.

3. Wait for a few minutes and verify that the deployment is deleted. To confirm the deployment is deleted, log in to the on-premises environment and navigate to the Service Fabric Explorer.

   The following applications should be deleted:

   - AXBootstapperAppType
- AXSFType
- FinancialReportingType
- RTGatewayAppType
- ReportingService

The following on-premises service fabric agent applications should not be deleted:

- LocalAgentType
- MonitoringAgentAppType

4. After all of the applications in step 3 are deleted, go back to LCS and click **Configure**.

5. Select the new topology for your platform.

6. Enter the environment name. You can use the same name or enter a new one.

7. Click **Advanced Settings**. You can now use the relevant configurations from the .json file that you saved to configure your environment.
This topic provides the detailed process for upgrading on-premises environments of Finance and Operations from version 7.x to 10.0.x.

**NOTE**

Please perform the upgrade with your sandbox environment before upgrading your production environment.

### On-premises upgrade from version 7.x to 10.0.x

**NOTE**

Be aware that this upgrade process takes time to complete and Finance and Operations will be inaccessible for the entire duration of the data upgrade.

To upgrade from version 7.x to 10.0.x, there are two possible paths that are currently supported.

An overview of each path is given below:

- **Upgrading from within VHD** - This path involves copying your database into the virtual hard disk (VHD) and executing the upgrade inside it. Overall, this is the simpler method.

- **Upgrading with VHD pointing to your database** - This path involves pointing the VHD upgrade process to your database. The upgrade process is still executed from within the VHD.

**NOTE**

The VHD does not need external network access in order to carry out the upgrade process.

### Prerequisites

1. In Lifecycle Services (LCS), go to the Shared Assets Library (right side of the screen).

2. Under Select asset type, choose Downloadable VHD, and download all parts of the VHD package that closely matches the version you will be upgrading to in your on-premises environment. The image requires a high amount of disk space, so be sure to download and extract on a drive with adequate free space.

3. The files that you downloaded are a self-extracting zip file. Extract the VHD to a location with a good amount of free space.

4. Using Hyper-V, launch a virtual machine (VM) and attach the VHD. (Note that the machine must be Generation 1.)

5. Connect to the VM. You can find the credentials in Running the Virtual Machine (VM) locally.

6. Depending on your planned on-premises target version of 10.0.x and the VHD image you downloaded, you may need to download and apply the required Application and Platform Update from the Shared
Asset Library under **Select asset type** and **Software deployable package**. For more information, see **Install deployable packages from the command line**.

7. If you have any extensions or customizations install them into the VHD now, otherwise the upgrade process will remove any data related to customizations. Check with your independent software vendor (ISV) or value-added reseller (VAR) if you need to prepare your environment before the upgrade.

**Upgrading from within VHD**

1. Shut-down on-premises AOS, BI, and MR servers or stop the Service Fabric Host Service in each of the nodes and set to disabled.

2. Back up your database from your on-premises environment (typically AXDB). For more information, see [Create a Full Database Backup](#).

3. In the VHD, go to C:\AOSService\PackagesLocalDirectory\Bin\CustomDeployablePackage and copy the MinorVersionDataUpgrade zip file.

4. Paste the file wherever you want and unzip it. For example: c:\D365FFOUnder\Bin\CustomDeployablePackage and copy the MinorVersionDataUpgrade zip file.

5. Open a Command Prompt as Administrator and change the directory to the unzipped folder in step 4.

6. Restore the backup that you created into the OneBox VM. For more information, see [Restore a Database Backup Using SSMS](#).

7. Optional: If the name of your restored database is not AXDB, using PowerShell with administrator privileges, execute:

   ```powershell
   .\Configure-On-Premises-Upgrade.ps1 -DatabaseName '<DB-name>'
   ```

   **NOTE**
   Substitute `<DB-Name>` with the appropriate value in your case (for example, AXDB). If you would like to edit more values, refer to the appendix of this topic.

   The script will run a database connection test to check that the information you provide is valid.

8. Using the Command Prompt from step 5, execute the following commands:

   a. `AxUpdateInstaller.exe generate -runbookid=upgrade -runbookfile=upgrade.xml -topologyfile=defaulttopologydata.xml -servicemodelfile=defaultservicemodeldata.xml`

   b. `AxUpdateInstaller.exe import -runbookfile=upgrade.xml`

   c. `AxUpdateInstaller.exe execute -runbookid=upgrade`

   During the execution of cleanup for data upgrade you may encounter an error:

   ```
   Stack trace: Call to TTSCOMMIT without first calling TTSEXECUTE.
   ```

   To resolve this, re-run the step with this command:

   ```powershell
   AxUpdateInstaller.exe execute -runbookid=upgrade -rerunstep=<failed-step>
   ```

9. When the upgrade process has finished successfully, back up the newly upgraded database. If you have customizations from ISVs or VARs, check if you have to run some post data upgrade scripts.

10. Restore the database into your environment with a different name from the production one (for example, AXDBupgraded).
11. Start on-premises AOS, BI, and MR servers, or start the services from the Service Fabric portal.

12. In LCS, open the project, and then, in the **Environments** section, delete the deployment. The applications should start to disappear from Service Fabric Explorer in the environment. This process may take longer depending on the number of nodes that you have. Check the service fabric explorer to verify that all applications have been deleted before deploying a new environment. Note that LCS might indicate that the environment is deleted before the actual process is finished.

13. If you had customizations:
   a. In LCS, go to the Shared Assets Library.
   b. Under **Select asset type**, choose **Model** and download: Dynamics 365 for Finance and Operations on-premises, Version 10.0.x Demo Data. Select the version closest to the 10.0.x environment that you will deploy as the on-premises baseline.
   c. Use this file to create a new database (typically AXDB) using the restore backup option from SQL server. For more information, see **Restore a Database Backup Using SSMS**.
   d. The database will need to be configured. Follow the steps in **Configure the Finance and Operations database**.
   e. In LCS, set up a new environment and deploy it with version 10.0.x (Redeploy). For more information, see **Set up and deploy on-premises environments (Platform update 12 and later)**. When you deploy, the database that you specify should be the one created in step 13c (typically AXDB).
   f. Apply your own customizations as well as ISV/VAR modules, to your newly created 10.0.x environment. Otherwise, when the environment initially syncs with the database it will delete any customization or extensions related data.
   g. Shut-down on-premises AOS, BI, and MR servers, or stop the services from the Service Fabric portal.
   h. Rename or delete the demo database (typically AXDB) used in the deploy and then rename your new database (typically AXDBupgraded) to the name that the demo database had (typically AXDB).
   i. Start on-premises AOS, BI, and MR servers, or start the services from the Service Fabric portal.

14. If you didn’t have customizations:
   a. (Optional) Rename your old database (typically AXDBold) and then rename your new database (typically AXDB). Make sure that in the next step you input the name of the upgraded DB.
   b. In LCS, set up a new environment and deploy it with version 10.0.x (Redeploy). For more information, see **Set up and deploy on-premises environments (Platform update 12 and later)**.

15. (Optional) If deployment fails because the financial reporting module failed, on the database that you are using for the new environment (typically AXDB), run the following command:

   ```sql
   ALTER TABLE RETAILTERMINALTABLE ADD CONSTRAINT PK_RecId PRIMARY KEY CLUSTERED (RECID)
   ```

**Upgrading with VHD pointing to your database**

1. Shut-down on-premises AOS, BI, and MR servers, or stop the services from the Service Fabric portal.

2. Back up your database from your on-premises environment (typically AXDB). For more information, see **Create a Full Database Backup (SQL Server)**.

3. Restore the backup that you just created into the database server and give it a different name (AXDBtoupgrade). For more information, see **Restore a Database Backup Using SSMS**.
4. Once connected, go to C:\AOSService\PackagesLocalDirectory\Bin\CustomDeployablePackage and copy the MinorVersionDataUpgrade zip file.

5. Paste the file wherever you want and unzip it. For example: C:\D365FFOUUpgrade\

6. Open a Command Prompt as Administrator and change the directory to the unzipped folder from step 5.

7. Open a new PowerShell as Administrator and execute:

```
.\Configure-On-Premises-Upgrade.ps1 -DatabaseName '<DB-name>' -DatabaseServer '<SqlServerName>' -
DatabaseUser '<User>' -DatabasePassword '<Password>'
```

**NOTE**
Substitute <*> with the values you require.

**NOTE**
- Only SQL Server authentication is officially supported for this upgrade. For more information, see Create a Database User.
- You will need to add the Certificate Authority certificate that signed your SQL Server certificate to the OneBox trusted certificate authorities. For more information, see Installing the trusted root certificate.
- Make sure the database user you use has the sysadmin server role assigned or at least All Privileges on the database you want to upgrade and has permissions to access tempDB. Step 6 of the upgrade process will fail if this is not true.
- When you install the Certificate Authority in the OneBox, make sure you use the FQDN or IP for connecting to the database that appears there. If you can't access it by using the domain name because it doesn't point to that server, edit your hosts file and add the DN and the IP it should resolve to.

8. Using the Command Prompt from step 6, execute the following commands:

a. `AxUpdateInstaller.exe generate -runbookid=upgrade -runbookfile=upgrade.xml -
topologyfile=defaulttopologydata.xml -servicemodelfile=defaultservicemodeldata.xml`

b. `AxUpdateInstaller.exe import -runbookfile=upgrade.xml`

c. `AxUpdateInstaller.exe execute -runbookid=upgrade`

During the execution of Cleanup for data upgrade you may encounter an error:

```
Stack trace: Call to TTSCOMMIT without first calling TTSBEGIN.\' on category \'Error\'.
```

To resolve this, re-run the step with this command:

```
AxUpdateInstaller.exe execute -runbookid=upgrade -rerunstep=<failed-step>
```

9. If you have customizations from ISVs or VARs, verify if you have to run some post data upgrade scripts.

10. Start on-premises AOS, BI, and MR servers, or start the services from the Service Fabric portal.

11. In LCS, open the project, and then, in the **Environments** section, delete the deployment. The applications should start to disappear from Service Fabric Explorer in the environment. This process may take longer depending on the number of nodes you have.

12. If you had customizations:

a. In LCS, go to the Shared Assets Library.
b. Under Select asset type, choose Model and download: Dynamics 365 for Finance and Operations on-premises, Version 10.0.x Demo Data. Select the version closest to the 10.0.x environment that you will deploy as the on-premises baseline.

c. Use this file to create a new database (typically AXDB) using the restore backup option from SQL server. For more information, see Restore a Database Backup Using SSMS.

d. The database will need to be configured. Follow the steps under Configure the Finance + Operations database.

e. In LCS, set up a new environment and deploy it with version 10.0.x (Redeploy). For more information, see Set up and deploy on-premises environments (Platform update 12 and later). When you deploy, the database that you should specify should be the one created in step 12c (typically AXDB).

f. Apply your own customizations as well as ISV/VAR modules, to your newly created 10.0.x environment. Otherwise when the environment initially syncs with the database it will delete any customization or extensions related data.

g. Shut-down on-premises AOS, BI, and MR servers, or stop the services from the Service Fabric portal.

h. Rename or delete the demo database (typically AXDB) used in the deploy and then rename your new database (typically AXDBupgraded) to the name the demo database had (typically AXDB).

i. Start on-premises AOS, BI, and MR servers, or start the services from the Service Fabric portal.

13. If you didn't have customizations:

a. (Optional) Rename your old database (typically AXDBold) and then rename your new database (typically AXDB). Make sure that in the next step you input the name of the upgraded database.

b. Set up a new environment and deploy it with version 10.0.x. For more information, see Set up and deploy on-premises environments (Platform update 12 and later).

14. (Optional) If deployment fails because the financial reporting module failed, on the database that you are using for the new environment (typically AXDB), run the following command:

```sql
ALTER TABLE RETAILTERMINALTABLE ADD CONSTRAINT PK_RecId PRIMARY KEY CLUSTERED (RECID)
```

Appendix

**Configure-On-Premises-Upgrade.ps1 usage**

**IMPORTANT**

This script is only meant to be run from a OneBox VHD environment.

The script requires that you pass at least the DatabaseName parameter. If you don’t pass it, the script will automatically request it.

If you want to pass an additional parameter like DatabaseServer, or DatabaseUser you can do so but this will cause the script to ask you for all additional parameters. This happens because the script will assume that you want to point the database connection to a machine outside the VM and those parameters will be required to correctly establish the connection.

The parameters that can be passed to the script are:

- **-DatabaseName** - Database name that you want to upgrade.
- **-DatabaseServer** - Database server containing Finance and Operations (on-premises) database.
- **-DatabaseUser** - Username for SQL Authentication.
- **-DatabasePassword** - Password for SQL Authentication.

After configuration has been passed, the script will execute a database connection test with the new parameters. If the script is unable to connect we recommend that you use Microsoft SQL Server Management Studio or some other tool to debug the connection from there.

**Configure-On-Premises-Upgrade.ps1**

```powershell
<#
.Synopsis
Configures a Onebox deployment to upgrade an OnPrem 7.x database to OnPrem 10.0.x

.DESCRIPTION
This must be executed before the upgrade process is carried out.

.EXAMPLE
\Configure-OnPremUpgrade.ps1 -DatabaseName 'AxDB'
\Configure-OnPremUpgrade.ps1 -DatabaseName 'AxDB' -DatabaseServer '127.0.0.1' -DatabaseUser 'axdbadmin' -DatabasePassword 'secretPass'
#>
[CmdletBinding()]
param
(
    # Database server containing Microsoft Dynamics 365 for Operations, on-premises database.
    [AllowNull()] [string] $DatabaseServer,
    # Database name that you want to upgrade.
    [Parameter(Mandatory = $true)] [string] $DatabaseName,
    # Username for SQL Authentication.
    [AllowNull()] [string] $DatabaseUser,
    # Password for SQL Authentication.
    [AllowNull()] [string] $DatabasePassword
)
$webroot = "C:\AOSsService\webroot"
$commandParameter = " -decrypt ""$webroot\web.config"
$command = Resolve-Path "$webroot\bin\Microsoft.Dynamics.AX.Framework.ConfigEncryptor.exe"
Start-Process $command $commandParameter -PassThru -Wait
if([string]::IsNullOrEmpty($DatabaseUser) -and [string]::IsNullOrEmpty($DatabasePassword) -and [string]::IsNullOrEmpty($DatabaseServer)) {
    [xml]$web = Get-Content $webroot\web.config
    $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.Database']").value = [string]$DatabaseName
} else {
    if([string]::IsNullOrEmpty($DatabaseServer)){
        $DatabaseServer = if($value = Read-Host 'What is the IP or FQDN of the Database server?' -valid) { [string]$value }
    }
```
$DatabaseServer = if($value = Read-Host 'What is the IP or FQDN of the Database server?) {'127.0.0.1'} else {'127.0.0.1'}

if([string]::IsNullOrEmpty($DatabaseUser)){

$DatabaseUser = if($value = Read-Host 'What is the SQL Authentication username? [axdbadmin]') {$value} else {'axdbadmin'}

}

if([string]::IsNullOrEmpty($DatabasePassword)){

$DBPassEn = if($value = Read-Host 'What is the SQL Authentication password?' -AsSecureString) {$value} else {''''}

$BSTR = [System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($DBPassEn)


}

[xml]$web = Get-Content $webroot\web.config


$web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.Database']").value = [string]$DatabaseName


} #Save Configuration to webroot config
$web.Save("$webroot\web.config")

#Reloading the configuration to run test
[xml]$web = Get-Content $webroot\web.config

$TestDbServer = $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.DbServer']").value
$TestDbName = $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.Database']").value
$TestDbUser = $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.SqlUser']").value
$TestDbPass = $web.SelectSingleNode("configuration/appSettings/add[@key='DataAccess.SqlPwd']").value

#Setting up connection test.
$DBConn = New-Object System.Data.SqlClient.SqlConnection
$DBConn.ConnectionString = "Data Source=$TestDbServer;User ID=$TestDbUser;Password='"$TestDbPass";Database=$TestDbName"

try{
    $DBConn.Open()

    $result = $true
}
catch{
    $result = $$.Exception.Message
}
\[
\text{\$result} = \_\_.\text{Exception}\text{.Message}
\]

\[
\text{Finally}{
\}
\]

\[
\text{\$dbConn.Close()}
\]

\[
\text{\$commandParameter} = " -encrypt \"webroot\web.config\""
\]

\[
\text{Start-Process \$command \$commandParameter -PassThru -Wait}
\]

\[
\text{if} (\text{\$result} \neq \text{\$true}){
\}
\]

\[
\text{Write-Host "\nThe connection to the Database Server failed:" -ForegroundColor Red}
\]

\[
\text{Write-Host \$result -ForegroundColor Red}
\]

\[
\text{else}{
\}
\]

\[
\text{Write-Host "\nThe connection to the Database Server was successful!" -ForegroundColor Green}
\]

\*

**Troubleshooting**

- Wrong database name or user doesn't have access to that database: Exception calling "Open" with "0" argument(s): "Cannot open database "AxDB1" requested by the login. The login failed. Login failed for user 'axdbadmin'."

- Could not establish a connection. Check ip/fqdn and ports: Exception calling "Open" with "0" argument(s): "A network-related or instance-specific error occurred while establishing a connection to SQL Server. The server was not found or was not accessible. Verify that the instance name is correct and that SQL Server is configured to allow remote connections. (provider: Named Pipes Provider, error: 40 - Could not open a connection to SQL Server)"

- Login credentials are not correct. Exception calling "Open" with "0" argument(s): "Login failed for user 'axdbadmin'."
This topic covers what updates you should expect to see and how you can get the latest updates using Lifecycle Services (LCS).

Get updates

To view available updates:

1. Sign in to LCS using your credentials.
2. In the LCS project, select an environment.
3. On the Environment page, scroll down to see the Available updates.

Types of updates

- **Binary updates** are pre-compiled and cumulative. Every subsequent binary update includes all previous updates. These updates don’t have to be compiled in a development environment, and they can be applied directly to a non-development environment from LCS.

  If you’re running an environment that has Commerce functionality and a customized instance of Cloud point of sale (POS), you must complete the additional steps that are listed under the SDK packaging. For Microsoft Dynamics 365 Commerce, all updates, even updates for application models, are released as binary updates.

  For all versions of Commerce and Finance and Operations apps that are version 8.1 and later, all updates, including updates for application models, are released as binary updates.

- **X++ updates** include updates to specific application functionality in application models. These updates can be independently downloaded and applied. You can select specific X++ updates to apply to your environment. Dependent X++ updates are automatically selected and downloaded. X++ updates are source code updates. Before they can be applied to a non-development environment, X++ updates must be compiled in a developer environment and merged with any customizations. X++ updates apply only to version 8.0 and earlier.

Update option by product and version

Based on your product and version, you will have different update options from Lifecycle Services.

**Finance and Operations apps**

- **Application version 8.1 and later (One Version)** - All updates for version 8.1 and later will have the One Version service update experience. It will be a cumulative, combined binary update of all of the application and platform updates. There will be no granular X++ updates starting with this release.

  Based on your environment version and the service update availability, you will have the option to choose the updates available to your environment. Each update option is associated with a version number and a build number.

  You may see one or more of the following update options.
<table>
<thead>
<tr>
<th>UPDATE</th>
<th>DESCRIPTION</th>
<th>AVAILABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality update</td>
<td>A quality update is a cumulative, roll-up build that contains fixes for known issues that are specific to the service update.</td>
<td>A quality update is available when your environment is running the same version of the current service update (n), or when your environment is running on one version older than the current service update (n-1). For example, if the current service update is version 10.0.2, you will have the option to choose a quality update if you’re running version 10.0.2, or if you’re running one version older, which is 10.0.1. There will be no quality update available for any version that’s older than 2 versions of the current service update. You will have to apply the latest service update to stay current.</td>
</tr>
<tr>
<td>Service update</td>
<td>A service update is the version currently automatically applied to customer environments based on the LCS project update settings. A service update is a cumulative, roll-up build that contains new features, functionality, and the related quality update that is generally available.</td>
<td>A service update is available if your environment has not been updated to the current service update version available for auto-update. Only the designated sandbox or production environment will be auto-updated if you have configured the update settings for the LCS project. However, you can manually apply the current service update version to other sandbox environments or your cloud-hosted environments.</td>
</tr>
<tr>
<td>Upcoming service update</td>
<td>An upcoming service update is the latest version that is generally available for self-update. An upcoming service update is a cumulative, roll-up build that contains new features, functionality, and the related quality update that is generally available.</td>
<td>An upcoming service update will be made generally available for self-deployment approximately 2 weeks prior to when Microsoft starts automatically applying this version based on your update settings for the LCS project.</td>
</tr>
</tbody>
</table>

- **Application version 7.3 with Platform update 4 and later** - This release will still have the granular X++ updates. Starting with Platform update 4, no overlayering is allowed on the platform modules, which means that the Platform binary updates tile is available to provide the platform updates as a cumulative update.

For customers that are on this combination, you will see the following tiles:

- **All X++ updates** - This tile shows all the granular X++ updates released by Microsoft.

- **Critical X++ updates** - This tile shows recommended KBs that are based on the telemetry data in your production environment. This tile will only show Production environments and a subset of the updates shown under the All X++ updates tile that are recommended for your environments.
- **All binary updates** - This tile shows a combined, cumulative binary update for both the Application and Platform.

- **Platform binary updates** - This tile shows only the Platform binary updates. If you want to update only the platform, you can get the update from this tile.

- **Application version 7.1, 7.2, 8.0, or earlier (except version 7.3) with Platform update 32 and earlier** - The product versions noted here are out of service. No new X++ updates are available. You can apply the X++ updates that have been released previously but no new X++ update will be published to LCS.

  Also, a platform update will not be available starting with Platform update 33. This means that you will not be able to apply the platform only update package if your application version is 7.1, 7.2, or 8.0 and earlier (except version 7.3). If you’re running any of these versions, you need to upgrade to the latest version to stay with the latest feature and functionality. For more information, see One Version service updates FAQ.

**NOTE**
If you are on a release that is noted above, you need to upgrade as soon as possible.

For the X++ updates that have been released for these versions, they are available from Issue Search in Lifecycle Services.

---

### Download binary updates

To download binary updates, follow these steps in LCS.

1. Select any of the binary update options, including Quality update, Service update, All binary updates, and Platform binary updates to view the combined list of application and platform binary updates.

2. On the **Binary updates** page, select **Save package**.

   **NOTE**
   You will not be able to select Knowledge Base (KB) articles to be saved because binary updates will automatically save all KBs in an update package.

3. On the **Review and save updates** page, select **Save package**.

4. In the **Save package to asset library**, enter the **Name** and **Description**, and select **Save package**.

5. Select **Done** to return to the environment page.

6. You’ll see the saved binary package in the asset library.

---

### Download X++ updates

To download X++ updates, follow these steps in LCS.

1. Select the **All X++ updates** tile to view the list of available application updates for an environment, or select the **Critical X++ updates** tile for the application updates that are recommended for your production environment.

2. On the **Add updates** page, select the applicable Knowledge Base (KB) numbers, and then select **Add** to
add selected KBs to the download package.

**NOTE**

For X++ updates, you can download all available updates at this point. Click **Select all**, and then select **Add** to add all KBs to the download package.

3. Select **Download package**.

4. On the **Review and download hotfixes** page, you can review the hotfixes that you selected, discard the package, return to the hotfix selections, or download the final hotfix package.

5. Download the package, and select **Done**.

**Additional resources**

- Apply updates to cloud environments
- Install metadata hotfixes in development environments
This topic describes how you can use Microsoft Dynamics Lifecycle Services (LCS) to automatically apply updates to cloud environments.

**IMPORTANT**

Updates are applied using deployable packages. Applying updates causes system downtime. All relevant services will be stopped, and you won't be able to use your environments while the package is being applied. You should plan accordingly.

**Supported environments**

All customer-managed and Microsoft-managed environments deployed through Lifecycle Services are supported. For more information about self-service environments, see [Update an environment](#).

**NOTE**

If you have a build environment, you can only use LCS to apply Binary updates and Data upgrade packages. You can't use LCS to apply an Application Deployable package.

For other environments (listed below), you must use Remote Desktop Protocol (RDP) to connect to the environment and install from the command line. For information about manual package deployment, see [Install deployable packages from the command line](#).

- Local development environments (Downloadable virtual hard disk [VHD])
- Multi-box dev/test environments in Microsoft Azure (Partner and trial projects)

**Key concepts**

Before you begin, you should understand deployable packages, runbooks, and the AXInstaller. A deployable package is a unit of deployment that can be applied in any environment. A deployable package can be a binary update to the platform or other runtime components, an updated application (AOT) package, or a new application (AOT) package. The AXInstaller creates a runbook that enables installing a package. For more details, see [Packages, runbooks, and the AXUpdateInstaller in depth](#) at the end of this topic.

**Supported package types**

- **AOT deployable package** – A deployable package that is generated from application metadata and source code. This deployable package is created in a development or build environment.
- **Application and Platform Binary update package** – A deployable package that contains dynamic-link libraries (DLLs) and other binaries and metadata that the platform and application depend on. This is a package released by Microsoft. This is available from the All binary updates tile from LCS.
- **Platform update package** – A deployable package that contains dynamic-link libraries (DLLs) and other binaries and metadata that the platform depend on. This is a package released by Microsoft. This is available from the Platform binary updates tile from LCS.
- **Commerce deployable package** – A combination of various packages that are generated after the Commerce code is combined.
**Merged package** – A package that is created by combining one package of each type. For example, you can merge one binary update package and one AOT package, or one AOT package and one Commerce deployable package. The packages are merged in the Asset library for the project in LCS.

**NOTE**

A binary package and a Commerce deployable package can't be included in the same merged package.

For information about how to download an update from LCS and what you see in the tiles based on your environment version, see [Download updates from Lifecycle Services (LCS)](https://docs.microsoft.com/en-us/lifecycle-services/download-updates).

If your environment is on an application version 8.1 and later, then the **Platform Update package** does not apply to your environment. Starting with 8.1 and later releases, **Application and Platform Binary update package** is the one that applies since application and platform will be combined into a single cumulative package and will be released by Microsoft. Also note that you will no longer be applying granular X++ hotfixes and will get all application and platform updates together. This means that on the environment details page, clicking on **View detailed version information** will not have details on the granular hotfixes or KBs applied as there is no way to apply them.

Prerequisite steps

- **Make sure that the package that should be applied is valid.** When a package is uploaded to the Asset library, it isn't analyzed. If you select the package, the package status appears in the right pane as **Not Validated**. A package must pass validation before it can be applied in an environment by using the following procedures. The status of the package will be updated in the Asset library to indicate whether the package is valid. We require validation to help ensure that production environments aren't affected by packages that don't meet the guidelines.

  There are three types of validations:
  - Basic package format validations
  - Platform version checks
  - Types of packages

- **Make sure that the package is applied in a sandbox environment before it's applied in the production environment.** To help ensure that the production environment is always in a good state, we want to make sure that the package is tested in a sandbox environment before it's applied in the production environment. Therefore, before you request that the package be applied in your production environment, make sure that it has been applied in your sandbox environment by using the automated flows.

- **If you want to apply multiple packages, create a merged package that can be applied first in a sandbox environment and then in the production environment.** Application of a single package in an average environment requires about 5 hours of downtime. To avoid additional hours of downtime when you must apply multiple packages, you can create a single combined package that contains one package of each type. If you select a binary package and an application deployable package in the Asset library, a **Merge** button becomes available on the toolbar. By clicking this button, you can merge the two packages into a single package and therefore reduce the total downtime by half.

- **Make sure that the application binary update package is applied to your dev/build environment AFTER it is applied to your sandbox and production environment.** If the application binary package is applied on your dev/build environment and this raises the platform build version to be higher than your target sandbox or production environment, you will be blocked from applying any AOT packages that are produced from this dev/build environment. To apply AOT packages produced from a dev/build environment, your dev/build instance must be equal to or lower than your target environments.
Apply a package to a non-production environment by using LCS

NOTE
For self-service type environments, see Update an environment.

Before you begin, verify that the deployable package has been uploaded to the Asset library in LCS.

1. For a binary update, upload the package directly to the Asset library. For information about how to download an update from LCS, see Download updates from Lifecycle Services (LCS). For an application (AOT) deployable package that results from an X++ hotfix, or from application customizations and extensions, create the deployable package in your development or build environment, and then upload it to the Asset library.
2. Open the Environment details view for the environment where you want to apply the update.
3. Click Maintain > Apply updates to apply an update.
4. Select the package to apply. Use the filter at the top to find your package.
5. Click Apply. Notice that the status in the upper-right corner of the Environment details view changes from Queued to In Progress, and an Environment updates section now shows the progress of the package. You can refresh the page to check the status.
6. Continue to refresh the page to see the status updates for the package application request. When the package has been applied, the environment status changes to Deployed, and the servicing status changes to Completed.

Apply a package to a production environment by using LCS

In a production environment, customers can schedule a downtime for when they want the update to be applied. For self-service type environments, see Update an environment.

IMPORTANT
An important prerequisite for applying a package to a production environment is that the package must be successfully applied to at least one sandbox environment in the same project.

1. After the update is successfully applied in a sandbox environment, go to the project’s asset library. On the Asset library page, select the Software deployable package tab, select the package that you want to move to production, and click Release candidate. This indicates that this package is ready for production deployment.
2. Open the Environment details view for the production environment where you want to apply the package.
3. Select Maintain > Apply updates to apply the package.
4. Select the package to apply in your production environment, and then click Schedule to submit a request to apply it.

NOTE
The list of packages includes only the packages that have been successfully signed off in the sandbox environment, and that have been marked as release candidates.

5. Specify the date and time to schedule the package application. Click Submit, and then click OK to confirm. Note that your environments will be unavailable to perform business while the package is being
6. At the scheduled downtime, package deployment will start.

7. After the environment is serviced, you can monitor the status. The **Servicing status** field indicates the status of package application. Additionally, a progress indicator shows the number of steps that have been run, out of the total number of steps that are available.

8. After the deployment is successfully completed, the **Servicing status** field is set to **Completed**.

9. If package application isn’t successfully completed, Microsoft will investigate the issue. The **Servicing status** field will indicate that package application has failed. The environment will be rolled back to a good state.

**Troubleshoot package deployment failures**

If package deployment fails, see **Troubleshoot package application issues**.

**Applying updates and extensions**

If you are updating a Tier-2 Sandbox or Production environment on application version 8.1.2.x or newer and have initialized Cloud Scale Unit, you will also need to update Commerce channel components. For more information, see **Update Retail Cloud Scale Unit**.

If you're using components (such as Modern POS), after you've applied updates and extensions in your environment, you must also update your in-store components. For more information, see **Configure, install, and activate Modern POS (MPOS)**.

**Packages, runbooks, and the AXUpdateInstaller in depth**

Deployable packages, runbooks, and the AXUpdateInstaller are the tools you use to apply updates.

**Deployable package** – A deployable package is a unit of deployment that can be applied in an environment. A deployable package can be a binary update to the platform or other runtime components, an updated application (AOT) package, or a new application (AOT) package. Deployable packages downloaded from LCS or created in a development environment cannot be applied across product types. For example, a Finance deployable package cannot be applied in a Commerce app environment, and vice versa. If you have an existing customization for a Finance and Operations app that is compatible with the Commerce app, and you would like to apply it to a Commerce environment, you will need to re-package your source code in a Commerce development environment, and conversely if moving in the other direction.
Runbook – The deployment runbook is a series of steps that are generated in order to apply the deployable package to the target environment. Some steps are automated, and some steps are manual. AXUpdateInstaller lets you run these steps one at a time and in the correct order.

- Generated based on topology of deployments with multiples VMs
- Contains step by step information for applying deployable package
- Provides sequence of steps across VMs in multi-box/ HA environment
- Integration for apply automation scripts at each step
  - Stop/start AOS service, batch service
  - Report deployment, DB sync, ...

AXUpdateInstaller – When you create a customization package from Microsoft Visual Studio or a Microsoft binary update, the installer executable is bundled together with the deployable package. The installer generates the runbook for the specified topology. The installer can also run steps in order, according to the runbook for a specific topology.

Additional resources
Install deployable packages from the command line
This topic will guide you through installing an Application Metadata hotfix on your development environment.

A metadata hotfix package contains changes (metadata or X++ source code) to model elements (XML files) in your development environment. A hotfix can also contain new model elements. A metadata hotfix package is in the form of an SCDP file. This article describes the process for installing a metadata hotfix package and explains how to share the package with other developers who are working on the same project.

Overall flow

The following diagram shows the overall flow.

Download the hotfix from LCS

For instructions about how to download a hotfix, see Download updates from Lifecycle Services (LCS). After you download the zip file, extract the SCDP metadata hotfix package from it, and put it in a local folder.

Install the hotfix

Before you begin

- This topic assumes that your packages folder is located at c:\AOSService\PackagesLocalDirectory\Bin. On some virtual machines (VMs), it might be located at c:\Packages, i:\AOSService\PackagesLocalDirectory\Bin, or k:\AOSService\PackagesLocalDirectory\Bin.
- If you're not using Microsoft Azure DevOps or another source control system, create a backup of your packages folder (which is also known as the metadata store). We don't recommend that you do development unless you use Azure DevOps.
- If you have Azure DevOps or Microsoft Team Foundation Server (TFS) version control, make sure that there are no files in the Pending Changes list of your current workspace. If you have pending changes, we recommend that you submit them or shelve them before you install the metadata hotfix.
Install the metadata hotfix package

To invoke the installation of the metadata hotfix, you can call the SCDPBundleInstall.exe utility from a command prompt. SCDPBundleInstall.exe is located in your packages bin folder.

Without version control (not recommended)

If you're not using Azure DevOps or TFS for source control, use the following command.

```
SCDPBundleInstall.exe -install -packagepath=<scdp file containing the hotfix> -metadatastorepath=<metadata packages root folder>
```

With version control (recommended)

If you're using Azure DevOps or TFS for source control, follow the steps below: Prepare the installation of the hotfix package using the command below. This step is not available if you are using a platform that is older than Platform update 2 (August 2016))

```
SCDPBundleInstall.exe -prepare -packagepath=<scdp file containing the hotfixes> -metadatastorepath=<metadata packages root folder> -tfsworkspacepath=<path of local workspace folder> -tfsprojecturi=<URI of the Azure DevOps or TFS project collection>
```

This will create a changeset of all the existing files on your environment that will be modified by the hotfix package, the prepare command will not install the hotfixes. Here is an example.

```
SCDPBundleInstall.exe -prepare -packagepath=c:\temp\hotfixbundle1234.axscdppkg -metadatastorepath= c:\AOSService\PackagesLocalDirectory -tfsworkspacepath= c:\AOSService\PackagesLocalDirectory - tfsprojecturi=https://myaccount.visualstudio.com/defaultcollection
```

Check-in your pending changes to create a backup of these files in your version control system. This will enable rolling back the hotfixes if needed. Install the hotfix package using the command below.

```
SCDPBundleInstall.exe -install -packagepath=<scdp file containing the hotfixes> -metadatastorepath=<metadata packages root folder> -tfsworkspacepath=<path of local workspace folder> -tfsprojecturi=<URI of the Azure DevOps or TFS project collection>
```

If you are using a platform that is older than Platform update 2 (August 2016), you do not need to specify the -install option. Here is an example.

```
SCDPBundleInstall.exe -install -packagepath=c:\temp\hotfixbundle1234.axscdppkg -metadatastorepath= c:\AOSService\PackagesLocalDirectory -tfsworkspacepath= c:\AOSService\PackagesLocalDirectory - tfsprojecturi=https://myaccount.visualstudio.com/defaultcollection
```

Azure DevOps/TFS parameters let you add the files that are modified by the package to your list of pending changes in Team Explorer.

**Required parameters**

```
/packagepath=[Path of the local scdp file containing the hotfixes downloaded from Lifecycle Service (LCS)]
/metadatastorepath=[Path of the local metadata store folder, such as c:\AOSService\PackagesLocalDirectory]
```

**TFS parameters**

If you're using Azure DevOps or TFS for source control, you should specify the following two parameters.
After the install command is invoked, the package installation process begins. As part of the installation process, some XML files in your metadata store folder will be updated to reflect the changes that were made in the fix itself. If you’re using Azure DevOps or TFS, these files will be added to the list of included changes in the **Pending Changes** window in Team Explorer.

Resolve conflicts that are generated by the installation of the hotfix

Sometimes, a metadata hotfix package contains changes to objects that have been customized in higher-layer models. In this case, the installation process automatically generates conflicts that must be resolved after the hotfix has been installed. The development tools let you create a project that groups all items that have conflicts. For example, if you have a VAR layer model in the Application Suite package that customizes the VendTable form, and you install a hotfix that modifies the VendTable form in the Sys layer model, conflicts might occur in your VAR layer model.

1. Click **Dynamics 365** > **Addins** > **Create project from conflicts**.
2. In the dialog box, select a model to check for conflicts.
3. Click **Create project**. A project is generated that contains only those elements in the selected model that...
were found to have conflicts after the hotfix was applied.

4. Open the designer for the conflicting element to view conflicts, and resolve them by using the tools that are provided.

Build and test on a local VM

Build all models that are affected by the hotfix, and test your application.

Check pending changes in to version control

When you're satisfied with all changes that are related to this update, check in your pending changes to Azure DevOps by using Team Explorer in Microsoft Visual Studio. Enter a comment in the Comment field, and then click Check In. A history of the changes is preserved in your source code repository.

Synchronize other development VMs

After a hotfix has been installed on a development VM as described in this article, you don't have to reinstall, resolve conflicts, and validate on other development VMs that are connected to the same Azure DevOps project. Developers and testers who are connected to the same Azure DevOps project can just synchronize the changes into their local VM and then build.

Deploy

After you've applied a metadata hotfix to your development environment, resolved conflicts, and validated your changes, you must create a deployable package and apply your changes to your test or sandbox environment. If you use a build instance for build and test automation, the build process will automatically create the deployable package for you. For more information, see Create deployable packages of models.
Patch SQL Server Reporting Services (SSRS) in one-box environments

11/24/2021 • 3 minutes to read • Edit Online

The following procedure is for one-box development environments only.

Patch the Reporting Service

The following procedure is for one-box development environments only.

- Download the patch .zip file from Lifecycle Services (LCS).
- If there are any font files in the Reporting Service patch’s data folder, install these to the machine where SQL Server Reporting Services (SSRS) is running. For more information about installing fonts on Windows, see How to install or remove a font in Windows. Any fonts that have already been installed do not need to be installed again.
- Copy the files in the Reporting Services patch scripts folder to the Report plug-in folder located under C:\Packages\Plugins\AxReportVmRoleStartupTask.
- Change the directory to the Report plug-in folder where you stored the script files.
- Using one of the methods listed below, replace the old instance of reporting extensions.
  - Remove/reinstall the reporting extension. The remove/reinstall option requires that redeploy all reports after you have finished the reinstallation.
  - Manually copy binaries to the SQL server binary folder. If you choose to manually copy the files, then you do not need to redeploy reports.

Remove/reinstall the reporting extension

Complete the following procedure as a user in the administrator group for the machine where SSRS is running.

- Using Windows PowerShell, remove the Dynamics SSRS extension by running the following script:
  - PowerShell .\DeploySsrsExtension.ps1 –UninstallOnly
- In PowerShell, reinstall the Dynamics SSRS extension by running the following script:
  - PowerShell .\DeploySsrsExtension.ps1
- Removing the reporting extension removes all the reports. If you have removed and then reinstalled the reporting extension, it is necessary to re-deploy the reports by running the following script:
  - Powershell .\DeployAllReportsToSsrs.ps1
- This task will take 20 to 30 minutes to complete.

Manually copy binaries to the SQL Server binary folder

1. Stop SQL Server Reporting Services. This can be done either from the Services management console or from the Reporting Services Configuration Manager.
2. Find the SQL Server Reporting Services binary folder. This folder is usually located at C:\Program Files\Microsoft SQL Server\MSRS11.MSSQLSERVER\Reporting Services\ReportServer\bin.

3. If any of the following files are in the patch, copy them to the SQL Server Reporting Services bin folder: * *

**NOTE**

Patches can either be full patches, which would contain all of the files used by the service, or incremental patches, which contain only the files that have changed. If you have an incremental patch, then some files may not be included. Files not included in the patch do not need to be replaced.

- Microsoft.Dynamics.Framework.ReportsExtensions.dll
- Microsoft.Dynamics.Framework.Reports.dll
- Microsoft.Dynamics.ApplicationPlatform.SSRSReportRuntime.man
- Microsoft.Dynamics.AX.Framework.Reports.Shared.dll
- Microsoft.Dynamics.AX.Framework.EncryptionEngine.dll
- Microsoft.Dynamics.AX.Framework.Utilities.dll
- Microsoft.Dynamics.AX.ReportConfiguration.axc
- Microsoft.WindowsAzure.ServiceRuntime.dll
- Microsoft.IdentityModel.dll
- msshrtmi.dll

Restart SQL Server Reporting Services.
**Reporting service installation**

The following changes are made with the reporting service installation: The following files will be copied into Reporting service bin folder (C:\Program Files\Microsoft SQL Server\MSRS11.MSSQLSERVER\Reporting Services\ReportServer\bin), and the corresponding SSRS config files will be updated so that SSRS is aware of the extension.

- Dynamics.Framework.ReportsExtensions.dll
- Dynamics.Framework.Reports.dll
- Dynamics.ApplicationPlatform.SSRSReportRuntime.Instrumentation.dll
- Dynamics.ApplicationPlatform.SSRSReportRuntime.man
- Dynamics.Platform.Integration.ClientSdk.Abstraction.dll
- Dynamics.AX.Framework.Reports.Sdk.Configuration.dll
- Dynamics.AX.Framework.EncryptionEngine.dll
- Dynamics.AX.Framework.Utilities.dll
- Dynamics.ApplicationPlatform.Environment.dll
- Dynamics.AX.ReportConfiguration.axc
- WindowsAzure.ServiceRuntime.dll
- IdentityModel.dll
- msshrtmi.dll

An SSRS service account will be updated to use the local system. A new SSRS catalog database DynamicsAxReportServer and temp database DynamicsAxReportServerTempDB database will be created, and SSRS will be configured to use these two databases. The default catalog database ReportServer and ReportServerTempDB still exist, but are set to not be used by reporting services. The SSRS service will be updated to use Windows Authentication. An xml configuration file ReportPVMConfiguration.xml will be created in the SSRS bin folder for the report runtime. A report root folder named Dynamics and a new security role named DynamicsBrowser will be created. Both AOS Web application AppPool identity and batch service account will be added to this custom role. Note that during deployment, the report folder will be deleted and then recreated. Therefore all the previously deployed reports will be deleted from the SSRS server. After you reinstall the reporting extension, you must redeploy the reports.
This topic explains how to update the development tools.

Use this tutorial to update your Visual Studio development tools with a new version. It explains how to uninstall your existing Visual Studio development tools and install the new extension. The new extension is in the form of an installable VSIX file. This file is a part of the binary hotfix available on the Dynamics Lifecycle Services (LCS) site. The VSIX file is located in the DevToolsService\Scripts folder of the binary hotfix package.

**NOTE**

You do not need to follow the instructions in this article if you are upgrading your Finance and Operations platform to Platform update 4 or newer. It is an automatic step that is part of the platform upgrade process.

Uninstall the existing Visual Studio extension

In order to install a new version of the development tools, you'll need to uninstall the existing version first. Verify the version of the development tools that you have installed. If you don't have it installed, you can skip this section.

**Verify your current version of the Visual Studio extension**

2. Select it and click OK.

**Uninstall the extension**

1. Open the Visual Studio Tools > Extensions and Updates dialog.
3. When the extension is uninstalled, exit Visual Studio.

**Install a new version of the extension**

1. Make sure Visual Studio is not running.
2. Double-click (or right-click and Open) the VSIX file of the new version.
3. Follow the installation instructions.
4. When installation is complete, you can start Visual Studio and start developing your application.
IMPORTANT
This topic is no longer updated. To see a current list of features that have been removed or deprecated from Finance and Operations apps, search for "Removed or deprecated features" content that relates to the app you're using.

This topic describes features that have been removed or deprecated from Dynamics 365 for Finance and Operations and previous releases of that product.

- A removed feature is no longer available in the product.
- A deprecated feature is not in active development and may be removed in a future update.

This list is intended to help you consider these removals and deprecations for your own planning.

Detailed information about objects in Finance and Operations apps can be found in the Technical reference reports. You can compare the different versions of these reports to learn about objects that have changed or been removed in each version of Finance and Operations apps.

**Finance 10.0.7 with Platform update 31**

**Chinese voucher types without Account groups selection**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Changed to the feature with account groups selection.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Application</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: By December 1, 2020, we plan to no longer support Chinese voucher types setup without Account groups selection. Find more details about new feature design in What's new in 10.0.7</td>
</tr>
</tbody>
</table>

**Finance and Operations 10.0.6 with Platform update 30**

**DimensionHash.getHash(str _message)**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Windows is deprecating the use of SHA1, as documented in Windows Enforcement of SHA1 Certificates.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Application</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: By April 1, 2020, developers must use the platform APIs found in the class HasFunction.</td>
</tr>
</tbody>
</table>

**Hash.ComputeSHA1Hash(string message)**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Windows is deprecating the use of SHA1, as documented in Windows Enforcement of SHA1 Certificates.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Platform</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: By April 1, 2020, developers must use the platform APIs found in the class HasFunction.</td>
</tr>
</tbody>
</table>

**FormDateTimeControl.setUtcString()**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>We are retiring the setUtcString() method, because a better replacement method is available.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Platform</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: By October 1, 2020, we plan to no longer support the setUtcString() method. Developers should be using the setUtcDateTime() method instead.</td>
</tr>
</tbody>
</table>

**Blocklist report (IT) – Feature reference IT-00001**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Not legally required.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Italian localization</td>
</tr>
</tbody>
</table>
### Domestic tax report – Feature reference IT-00003

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Not legally required.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Italian localization</td>
</tr>
</tbody>
</table>

| Status                        | Deprecated: By October 1, 2020, we plan to no longer support this report. |

<table>
<thead>
<tr>
<th>Deployment option</th>
<th>All</th>
</tr>
</thead>
</table>

### October 2019 deprecation announcement

**Flowchart diagrams in Business process modeler**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>We are deprecating the flowchart diagrams component in Business process modeler (BPM), because the legacy design caused low usage.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Areas affected</td>
<td>Business process modeler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Status</th>
<th>Deprecated: The flowchart diagrams component in BPM is expected to be removed in 2020. The following functionality will be unavailable:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- All flowcharts will be read-only and unavailable for editing. The shape properties that are associated with flowchart activities will also be unavailable. These flowcharts include both the default flowcharts that are automatically generated and customized flowcharts that are modified based on those default flowcharts.</td>
</tr>
<tr>
<td></td>
<td>- The process steps will be read-only and unavailable for editing.</td>
</tr>
<tr>
<td></td>
<td>- The legacy fit/gap analysis feature will be unavailable. Therefore, no gap list will be automatically created or available for export.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This feature had previously been deprecated and replaced by Microsoft Azure DevOps integrations.</td>
</tr>
<tr>
<td></td>
<td>- The version history of the flowchart will be unavailable.</td>
</tr>
</tbody>
</table>

Note: This feature had previously been deprecated and replaced by Microsoft Azure DevOps integrations.
### US Payroll tax updates

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>We are retiring tax updates for the US Payroll functionality due to low usage and enhanced functionality that is now offered via strategic integrations.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Payroll</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: By July 31, 2024, we plan to no longer provide tax updates to US Payroll customers. The functionality will remain in the product, but enhancements will no longer keep the functionality up to date, and any product defects will be evaluated on a case-by-case basis.</td>
</tr>
</tbody>
</table>

**NOTE**

This represents a change from the original discontinuation date of October 1, 2021. For more information, see [Tax updates being retired for US Payroll feature in Microsoft Dynamics 365 for Finance and Operations](#).

### Data management staging clean up

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Does not meet the core requirements that are needed for scheduling periodic cleanup.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, the Job history cleanup feature is being added to meet the scenarios holistically.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Data management</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: Target timeframe for the functionality to be removed is December 2020.</td>
</tr>
</tbody>
</table>

### Finance and Operations 10.0.4 with Platform update 28

**France: FEC Accounting data export in XML**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Replaced by TXT format, French FEC audit file is available through General ledger &gt; Periodic tasks &gt; Data export.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Value</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Replaced by another feature?</td>
<td>Yes</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>General ledger</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated. Target timeframe for the functionality to be removed is July 2020.</td>
</tr>
</tbody>
</table>

### Legacy navigation bar

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason for deprecation/removal</td>
<td>Header alignment with other Dynamics and Office products. For more details, see Updated navigation bar that aligns with the Office header.</td>
</tr>
<tr>
<td>Replaced by another feature?</td>
<td>Starting in Platform update 24, a restyled navigation bar that features search was introduced.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Web client</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: Starting in April 2020, the legacy navigation bar will no longer be available. Until that point, customers can revert to the legacy navigation bar through the Client performance options page.</td>
</tr>
</tbody>
</table>

### Finance and Operations 10.0.2 with Platform update 26

#### Legacy default action behavior

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason for deprecation/removal</td>
<td>The legacy behavior for default actions in grids results in an unexpected column having the default action link after grid columns have been reordered via personalization. The new sticky default action feature corrects this. For more details, see Sticky default actions in grids.</td>
</tr>
<tr>
<td>Replaced by another feature?</td>
<td>Starting in Platform update 21, a feature for &quot;sticky default actions&quot; was introduced. This feature can be enabled on the Client performance options page.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Grids in the web client</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: Starting in April 2020, sticky default actions will be the default behavior, without a mechanism to revert to the legacy behavior.</td>
</tr>
</tbody>
</table>
### Legacy "is one of" filtering experience

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The &quot;is one of&quot; filtering experience went through a redesign in Platform update 22, with the plan for this to eventually be the only &quot;is one of&quot; filtering experience.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Starting in Platform update 22, an improved &quot;is one of&quot; filtering experience became available on the Client performance options page. For more information, see Optimized is one of filtering experience.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Web client</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: Starting in April 2020, the improved &quot;is one of&quot; experience will be the default behavior, without a mechanism to revert to the legacy behavior.</td>
</tr>
</tbody>
</table>

### Parameter to enable sales orders with multiple project contract funding sources

Support for creating project-based sales orders where the project contract has multiple funding sources is enabled with the Project management parameters setting Allow sales orders for project with multiple funding sources. By default, this parameter is not enabled.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The functionality will always be enabled after the parameter is removed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No. The functionality to support project-based sales orders with multiple funding sources will always be enabled.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>The Allow sales orders for projects with multiple funding sources parameter will be removed. The following methods will be modified when the parameter is removed: ctrlSalesOrderTable method in ProjStatusType class, validate method for ProjId field, and run method in SalescreateOrder form. The following methods will be deprecated when the parameter is removed: IsSalesOrderAllowedForMultipleFundingSources in ProjTable table file, IsAllowSalesOrdersForMultipleFundingSourcesParam Enabled method in ProjTable table file, AllowSalesOrdersForMultipleFundingSources data field in ProjParameters form and ProjParameterEntity files, IsAssociatedToMultipleFundingSourcesContract private method in ProjTable table file.</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecation is planned for the April 2020 release wave.</td>
</tr>
</tbody>
</table>

### Legacy workflow reports for tracking and instance status
<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The legacy workflow reports for tracking and instance status are being deprecated because they are no longer referenced from the navigation. The report names are WorkflowWorkflowInstanceByStatusReport and WorkflowWorkflowTrackingReport.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>The workflow history form can be used instead.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Web client</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: Target timeframe for the functionality to be removed is April 2020.</td>
</tr>
</tbody>
</table>

**Finance and Operations 10.0.1 with Platform update 25**

**Deprecated APIs and potential breaking changes**

**Deriving from internal classes is deprecated**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Before Platform update 25, it was possible to create a class or table that derives from an internal class/table that is defined in another package/module. This is not a safe coding practice. As of Platform update 25, the compiler will display a warning.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>The compiler warning will be replaced by an error in Platform update 26. This change is backward compatible at runtime, which means that Platform update 25 or newer can be deployed on any sandbox or production environment without the need to modify custom code. This change only affects development and compile time.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Visual Studio development tools</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: The warning will become a compilation error in Platform update 26.</td>
</tr>
</tbody>
</table>

**Overriding internal methods is deprecated**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Before Platform update 25, it was possible to override an internal method in a derived class that is defined in another package/module. This is not a safe coding practice. As of Platform update 25, the compiler will display a warning.</th>
</tr>
</thead>
</table>
## Replaced by another feature?

This warning will be replaced by a compile error in Platform update 26. This change is backward compatible at runtime, which means that Platform update 25 or newer can be deployed on any sandbox or production environment without the need to modify custom code. This change only affects development and compile time.

<table>
<thead>
<tr>
<th>Product areas affected</th>
<th>Visual Studio development tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: The warning will become a compilation error in Platform update 26.</td>
</tr>
</tbody>
</table>

### Finance and Operations 10.0.0 with Platform update 24

#### Renaming released products

**Reason for deprecation/removal**

When you use the **Rename primary key** function to change the ItemId of a released product, only direct foreign key references are updated. Any other references to the released product, such as from production orders, will retain the old ItemId. As a result, there could be inconsistent data that will eventually block business processes.

<table>
<thead>
<tr>
<th>Replaced by another feature?</th>
<th>No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product areas affected</td>
<td>Product information management</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Finance and Operations 10.0.0 with Platform update 24.</td>
</tr>
</tbody>
</table>

### Finance and Operations 8.1.3 with Platform update 23

**SQL Server Reporting Services ReportViewer Control**

Customers can use the **Export** action provided by the embedded SQL Server Reporting Services (SSRS) ReportViewer control to download documents produced by Finance and Operations applications. This HTML-based presentation of the report offers users a non-paginated preview of the document.

<p>| Reason for deprecation/removal | The non-paginated nature of the HTML-based preview experience does <strong>not</strong> deliver fidelity with the physical documents ultimately produced by Finance and Operations. By fully embracing PDF as the standard format for business documents, users are able to take advantage of a modern viewing experience with improved performance when producing application reports. |</p>
<table>
<thead>
<tr>
<th><strong>Replaced by another feature?</strong></th>
<th>Going forward, PDF documents will be the default format for reports rendered by Finance and Operations.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product areas affected</strong></td>
<td>This change does not impact customer scenarios where reports are distributed electronically or sent directly to printers.</td>
</tr>
<tr>
<td><strong>Deployment option</strong></td>
<td>All</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Deprecated: A removal date has not been set for this feature. The functionality to automatically preview application reports using an embedded PDF viewer is planned for the May 2019 Platform update.</td>
</tr>
</tbody>
</table>

**Client KPI controls**

Embedded key performance indicators (KPIs) could be modeled in Visual Studio by a developer and further customized by the end user.

<table>
<thead>
<tr>
<th><strong>Reason for deprecation/removal</strong></th>
<th>The native client controls used to define KPIs have low customer uptake and rely on a developer to add trackable metrics.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>PowerBI.com service delivers world-class tooling for defining and managing KPIs based on data from external sources. In an upcoming release, we plan to enable you to embed solutions hosted on PowerBI.com in application workspaces.</td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
<td>This update will prevent developers from introducing new KPI controls in Visual Studio designer.</td>
</tr>
<tr>
<td><strong>Deployment option</strong></td>
<td>All</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Deprecated APIs and future breaking changes**

**Field groups containing invalid field references**
### Reason for deprecation/removal

It is possible for table metadata definitions to have field groups containing invalid field references. If deployed, this can cause runtime failures in Financial Reporting and SQL Server Reporting Services (SSRS). This issue is currently categorized as a *compiler warning* rather than an *error*, meaning that the deployable package creation and deployment can proceed without fixing the issue. To fix this issue:

1. Remove the invalid field reference from the table field group definition.
2. Recompile.
3. Ensure any warnings or errors are addressed.

### Replaced by another feature?

This warning will be replaced by a compile error in the future.

### Product areas affected

Visual Studio development tools

### Deployment option

All

### Status

Deprecated: The warning is a compile-time error with platform updates for version 10.0.11 of Finance and Operations apps.

### Complete list

To access the full list of APIs that are being deprecated, see [Deprecation of methods and metadata elements](#).

## Finance and Operations 8.1 with Platform update 20

### Batch transfer rules for subledger journal account entries

The Synchronous transfer mode is being deprecated in the General ledger parameters. This mode is replaced by Asynchronous and scheduled batch only, which already exist as options for transfer. For additional information, see the [General Ledger Parameters – Batch transfer rules](#) blog.

### Reason for deprecation/removal

We are removing the synchronous option due to performance impact to the system.

### Replaced by another feature?

Asynchronous and scheduled batch are options to use in place of Synchronous.

### Product areas affected

General Ledger, Accounts payable, Accounts Receivable, Procurement, Expense

### Deployment option

All

### Status

Deprecated: Target timeframe for the functionality to be removed is the 10.0 version.

### Electronic reporting for Russia

Feature for configuring .txt and .xml file formats of declarations.
<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Replaced with Electronic reporting.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>General Ledger</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
</tbody>
</table>

Financial reports generator for Russia

A tool for setting up data collection for accounting and tax reports, and to export data to XLS and DOC report templates. Functional parts: Export data to XLS and DOC report templates, queries, fixed requisites are removed.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Removed parts are replaced with Electronic reporting.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes. Financial reports setup user interface should be used for setting up data collection rules by GL accounts or tax registers. Export data to various file types, fixed requisites and query-like data collection rules should be configured in Electronic reporting.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>General ledger.</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
</tbody>
</table>

Integration with external providers for sending electronic reporting through communication channels for Russia

Feature exporting generated electronic files of declarations to folder for further sending to official providers of electronic reporting as well as importing state back.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Replaced with electronic messages configurable feature.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>General Ledger, Tax</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
</tbody>
</table>

Profit tax register wizard
Feature for creating templates for new profit tax registers. This feature creates X++ objects for new registers, which are then created as templates with the appropriate calculation logic added in.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Feature is not compatible with the Finance and Operations extensibility model.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Tax</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
</tbody>
</table>

**Payroll and Human Resources for Russia**
Russian country specific module for managing staff administration information, timesheet details for employees, payroll accounting, and creating pay statements.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Payroll is not included in the global strategic focus of the Dynamics 365 portfolio. Partners and ISVs are best positioned to provide payroll functionality that is compliant with local regulations and tax updates.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Russian Payroll and Human Resources Management</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: Target timeframe for the functionality to be removed is one of future updates of the 10.0 version.</td>
</tr>
</tbody>
</table>

**Finance and Operations 8.0 with Platform update 15**
No features have been removed or deprecated with this release. Platform update 15 is cumulative and contains new or changed features from Platform update 13, Platform update 14, and Platform update 15.

**Finance and Operations, Enterprise edition 7.3 with Platform update 12**

**Personalized product recommendations**
Starting February 15, 2018, retailers will no longer be able to display personalized product recommendations on a point of sale (POS) device. For more information, see [Product recommendations overview](#).
We are removing the current version of the product recommendation service as we redesign this feature with a better algorithm and newer retail-oriented capabilities.

No. However, after Spring 2018, we plan to bring back this feature to leverage a new recommendation service.

Personalized product recommendations in POS.

All

Removed as of February 15, 2018. This affects customers running Dynamics 365 for Operations 1611 and later.

The possibility to introduce custom functions to be used in the ER expression builder (for more information, see Extend the list of Electronic reporting (ER) functions) is not supported any more. Due to changes of the ER APIs, the API to call built-in functions from the ER expression builder became internal and can’t be extended any longer.

None. Whenever a new built-in function is needed, a new extension request must be addressed to the ER framework team.

As a temporary work around while the requested function is under development by the ER team, the required logic can be programmed as a method of a custom application class. This method can be accessed in an ER expression as a property of the added ER data source of the Application\Class type that refers to that custom application class.

Electronic reporting framework

All

Removed as of Finance and Operations, Enterprise edition 7.3.

These two reports are no longer supported in Finance and Operations. Instead, the Inventory aging report can be used to improve the user experience.

Duplicate functionality

Yes. The two reports have been replaced by the Inventory aging report.
<table>
<thead>
<tr>
<th>Product areas affected</th>
<th>Inventory management, Cost management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: The menu items for the two reports have been removed in version 7.3. However, the code for the reports remains in the product. The plan is to remove the code in a future release.</td>
</tr>
</tbody>
</table>

**Power BI content packs available on AppSource**

The **Cost management**, **Financial performance**, and **Retail channel performance** content packs, available on the [Microsoft AppSource](https://appsource.microsoft.com) site, are deprecated as a consequence of product updates in Microsoft Power BI. System administration forms used to deploy these content packs to PowerBI.com are also being deprecated in Finance and Operations.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Product updates in Microsoft Power BI.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>The <strong>Cost management</strong>, <strong>Financial performance</strong>, and <strong>Retail channel performance</strong> content packs, available on the <a href="https://appsource.microsoft.com">AppSource</a> site, are being replaced by analytical applications which allow for solution integrations at the database level. For more information about analytical applications, see <a href="https://docs.microsoft.com/en-us/power-bi/service/embed-power-bi">Embedded Power BI in workspaces</a>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product areas affected</th>
<th>Cost management, Finance, and Retail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment option</td>
<td>Cloud only (Integration with PowerBI.com is not supported in on-premises deployments.)</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: Target timeframe for the functionality removal is Q2 2018.</td>
</tr>
</tbody>
</table>

**Standard UI in data management workspace**

The standard UI in data management is the legacy UI, which is the default UI presented to the users when they visit the data management workspace.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>We are investing in providing new user experiences in the new UI.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>The new UI called <em>Enhanced views</em> is replacing the old UI.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Data management workspace</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: Target timeframe for the functionality to be removed is Q2 2018.</td>
</tr>
</tbody>
</table>
**Excise, Sales Tax, Service Tax for India**

These taxes have been subsumed into Indian GST.

<table>
<thead>
<tr>
<th>Reason for removal or deprecation</th>
<th>These taxes have been subsumed into Indian GST.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Indian GST</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Tax</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**File Validation Utility (FVU) for India**

<table>
<thead>
<tr>
<th>Reason for removal or deprecation</th>
<th>Lack of customer usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Indian withholding tax</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**TDS/TCS certificate for India**

Users can download this from the government portal.

<table>
<thead>
<tr>
<th>Reason for removal or deprecation</th>
<th>Lack of customer usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Indian withholding tax</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Export/import (EXIM) incentive scheme for India**

<table>
<thead>
<tr>
<th>Reason for removal or deprecation</th>
<th>Lack of customer usage</th>
</tr>
</thead>
</table>
### Dynamics 365 for Retail 7.2

**Personalized product recommendations**

Starting February 15, 2018, retailers will no longer be able to display personalized product recommendations on a point of sale (POS) device. For more information, see [Product recommendations overview](#).

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>We are removing the current version of the product recommendation service as we redesign this feature with a better algorithm and newer retail-oriented capabilities.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No. However, after Spring 2018, we plan to bring back this feature to leverage a new recommendation service.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Personalized product recommendations in POS.</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of February 15, 2018. This affects customers running Dynamics 365 for Retail 7.2 and later.</td>
</tr>
</tbody>
</table>

### Finance and Operations, Enterprise edition July 2017 with Platform update 8

**Currency conversion for accounting and reporting currencies**

Currency conversion for accounting and reporting currencies was introduced when the euro was introduced.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Limited usage and addition of the Copy legal entity functionality as a replacement.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No, but the Copy legal entity and Configurations features were added to make it easier to move to a company that has changing core requirements.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Financial management</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>
### Warehouse mobile devices portal

Warehouse mobile devices portal (WMDP) was a standalone component that was intended for on-premises self-deployment. This component is no longer supported in Finance and Operations. A native app that improves the user experience has replaced the functionality of WMDP.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Duplicate functionality.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes. This feature has been replaced by Finance and Operations - Warehousing. For more information about setup and prerequisites, see <a href="#">Install and configure the Warehousing app overview</a>.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Warehouse management, Transportation management</td>
</tr>
<tr>
<td>Deployment option</td>
<td>Warehouse mobile devices portal (WMDP) was a standalone component that was intended for on-premises self-deployment.</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: Target timeframe for the functionality to be removed is Q4 2019.</td>
</tr>
</tbody>
</table>

### Advanced bank reconciliation matching rule for manual matching

A matching rule was used to select and mark a bank document when documents were manually matched in the reconciliation worksheet.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Limited usage.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No. Column filtering capabilities should be used to find documents for reconciliation.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Cash and bank management</td>
</tr>
<tr>
<td>Deployment option</td>
<td>All</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of July 2017.</td>
</tr>
</tbody>
</table>

### Dynamics 365 for Operations 1611 with Platform update 3

#### AEB payment formats for Spain

The Consejo Superior Bancario payment formats were used to send remittance files to the bank for customer payments and vendor payments. The content of these formats was determined by the Asociación Española de Banca. It covers Cuaderno 19, 32, 58, 34.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment formats are no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes. ISO20022 Credit transfer and Direct debit payment formats for Spain</td>
</tr>
</tbody>
</table>
### Product areas affected
Accounts payable, Accounts receivable

### Status
Deprecated: A removal date has not been set for this feature.

### Bank payments transfer for Lithuania
Bank payment transfers were generated and printed by using the Payment transfer (LT) export format for Lithuania. The Lithuanian market began to use LITAS, the unified electronic banking system, in 2005.

- **Reason for deprecation/removal**
  The payment formats are no longer used.

- **Replaced by another feature?**
  Yes, ISO20022 Credit transfer payment format for Lithuania

### Bank payments transfer for Lithuania

- **Product areas affected**
  Accounts payable

- **Status**
  Deprecated: A removal date has not been set for this feature.

### BBS Direkte Remittering payment formats for Norway
BBS Direkte Remittering payment formats include customer payment collection export (direct debit) and return message import.

- **Reason for deprecation/removal**
  The payment formats are no longer used.

- **Replaced by another feature?**
  The AvtaleGiro customer payment format for Norway can be used to generate direct debit messages. Return message import will be implemented in future releases.

### BBS Direkte Remittering payment formats for Norway

- **Product areas affected**
  Accounts payable, Accounts receivable

- **Status**
  Deprecated: A removal date has not been set for this feature.

### Chart of Accounts tool for Spain
This tool is used when a chart of accounts in Spain requires major changes. Users can import a new chart of accounts in Microsoft Excel or text format, and can also import financial statements.

- **Reason for deprecation/removal**
  Limited usage

- **Replaced by another feature?**
  No

### Chart of Accounts tool for Spain

- **Product areas affected**
  General ledger

- **Status**
  Deprecated: A removal date has not been set for this feature.
Dom80 payment format for Belgium
Legacy Belgian payment format for payment collection (direct debit).

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment format is no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, ISO 20022 Direct debit payment format for Belgium</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts receivable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

DTA/EZAG payment formats for Switzerland
DTA/EZAG formats are integrated into the ESR system, because they can carry on the reference number. Because the reference number isn’t mandatory, these formats can be used to process any vendor payments. These formats are used by companies that have a bank account in a location other than “Postfinance.”

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment formats are no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, ISO20022 Credit transfer payment format for Switzerland</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

EDIFACT-DIRDEB payment format for Austria
EDIFACT-DIRDEB payment format for payment collection (direct debit).

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment format is no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, ISO 20022 Direct debit payment format for Austria</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts receivable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

EDIVAT for Belgium
EDIVAT is an obsolete Belgian standard for electronic declaration via secure mail. Dynamics AX 2012 retains the read-only solution to enable access to the historical data.

<p>| Reason for deprecation/removal | The functionality is no longer used. |</p>
<table>
<thead>
<tr>
<th>Replaced by another feature?</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product areas affected</td>
<td>General ledger</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

### eGiro EDIFACT CREMUL payment import format for Norway

eGiro is based on the international UN EDIFACT CREMUL (Multiple Credit Advice Message) standard that is used for automatic posting of customer payments. In Dynamics AX, eGiro is implemented as a customer payment import format.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment format is no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, the ISO20022 Camt.054 notification import.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts receivable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

### External inventory for Poland

Evidence of goods that are taken from a vendor for sales without purchase. Goods that are handled in external inventory don’t affect standard inventory, and can be sold and then purchased automatically. This process creates real inventory movements.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Replaced by another feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, the core Inbound consignment functionality</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable, Inventory management</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

### Financial reports generator for Eastern Europe

A tool is used to set up data collection for accounting and tax reports, and to export data to XLS and DOC report templates.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Limited usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No. The tool will be replaced by Electronic reporting configurations in future releases.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>General Ledger</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Import of customer payment transactions for Finland**

You can select an import format for Finnish payments to import customer payment transactions from an external file that the bank provides.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment format is no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, the ISO20022 Camt.054 notification import.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts receivable</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Import of payment transactions into a general ledger journal for Finland**

A format that is specific to Finland is used to import accounting transactions into the general ledger.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment format is no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, the ISO20022 Camt.053 bank statement import using Advanced Bank Reconciliation.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts receivable</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Integration with Isabel synchronized (CIS) for Belgium**

Isabel is the framework for electronic banking in Europe and is a de-facto standard in Belgium.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Integration with Isabel client has been discontinued.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No. The payment formats that are no longer used are replaced by ISO20022 Credit transfer payment format for Belgium.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>
### Modifications in the chart of accounts and accounting rules for Spain

This feature is used for changes in the chart of accounts and accounting rules in Spain. It maps accounts to help transform the old chart of accounts into the new chart of accounts, and compares the previous fiscal year with the new fiscal year, even if they were posted to different account numbers.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Limited usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>General ledger</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

### Pagamento Fornitori vendor payment format

Legacy Italian payment format for credit transfers.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment format is no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, ISO20022 Credit transfer payment format for Italy</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

### Payment export formats for Estonia

The Telehansa and Teleservice formats are used for bank payment export.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment formats are no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, ISO20022 Credit transfer payment format for Estonia</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

### Payment file archive for Norway

When payment files are generated, the file archive automatically archives all files that are created, even files that were previously written or read.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Replaced by another feature</th>
</tr>
</thead>
</table>
Replaced by another feature?

Yes, Electronic reporting archived jobs

Product areas affected

Accounts payable, Accounts receivable, Organization administration

Status

Deprecated: A removal date has not been set for this feature.

Payment import formats for Estonia

The Telehansa and TeleTeenus formats are used for bank payment import.

Reason for deprecation/removal

The payment formats are no longer used.

Replaced by another feature?

Yes, the ISO20022 Camt.054 bank notification import.

Product areas affected

Accounts receivable

Status

Deprecated: A removal date has not been set for this feature.

Payroll information in Human Resources

Human Resources Payroll information

Reason for deprecation/removal

This functionality has been replaced by core Payroll and Human Resources pages.

Replaced by another feature?

Benefits, Earnings, and other related pages that were previously in US Payroll have been reconfigured, and are now part of the core Human Resources configuration to help support external payroll processing. This functionality is accessed by using the Human Resources 1 > Payroll configuration key.

Product areas affected

Human Resources, Payroll

Status

Removed as of Dynamics 365 for Operations version 1611.

Performance management goal workflow

Performance management includes goal management and integration with performance reviews.

Reason for deprecation/removal

Performance management was redesigned, and the number of goal pages was reduced to simplify the process.

Replaced by another feature?

No. Goals are visible to managers through the Manager Self Service portal, and can be changed and viewed by the manager.
<table>
<thead>
<tr>
<th>Product areas affected</th>
<th>Human capital management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>Removed as of Dynamics 365 for Operations version 1611.</td>
</tr>
</tbody>
</table>

**Postgirot and Postgirot Utland payment formats for Sweden**
Postgirot and Postgirot Utland payment formats for Sweden.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The payment formats are no longer used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, ISO20022 Credit transfer payment format for Sweden</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Radio frequency identifier**
Radio Frequency Identification (RFID) is a data-collection technology that uses electronic tags to store identification data and a no-line-of-sight requirement reader to capture the identification data.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Low customer usage and a limited feature set.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Inventory management</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics 365 for Operations 1611.</td>
</tr>
</tbody>
</table>

**Report about state invoices numbering for Latvia**
Latvian legislation provides specific rules about the numbering of sales invoices. The functionality lets you assign specific numbers to sales invoices, based on the user or user group. You can then generate a report or an XML file. You can also print a report about invoice numbers that are used.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The state invoice numbering no longer has to be maintained. The report about used invoice numbers is no longer required.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts receivable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Set up the names of the manager and general accountant of a company for Lithuania**
The names of the manager and the general accountant of a company can be specified in the company information and used in different local report printouts.

### Shipping carrier interface

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Replaced by another feature?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason for deprecation/removal</td>
<td>Duplicate functionality</td>
</tr>
<tr>
<td>Replaced by another feature?</td>
<td>Partially replaced by Transportation management</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Sales and marketing, Inventory management</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics 365 for Operations version 1611.</td>
</tr>
</tbody>
</table>

### Telepay payment formats for Norway

Telepay payment formats include vendor payment export (credit transfer) and customer payment collection (direct debit).

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Replaced by another feature?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason for deprecation/removal</td>
<td>The payment formats are no longer used.</td>
</tr>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, ISO20022 Credit transfer payment format and AvtaleGiro customer payment format for Norway, as well as pain.002 and camt.054 bank notification return files import.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable, Accounts receivable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

### Vendor payment export formats for Finland

Two formats for exporting payments are available for Finland. LM02 (FI) is used for domestic payments, and LUM2 (FI) is used for foreign payments.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Replaced by another feature?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason for deprecation/removal</td>
<td>The payment formats are no longer used.</td>
</tr>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, ISO20022 Credit transfer payment format for Finland</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Warehouse management II**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The Warehouse management II solution (WMS II) that was available in the Inventory management module duplicates functionality that is in the Warehouse management module that was released in Dynamics AX 2012 R3.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>The Warehouse management module that was released in AX 2012 R3, Dynamics AX 2012 R3 CU8, and Dynamics AX 2012 R3 CU9 replaces the Warehouse management II features. The new module has more advanced features and more flexible warehouse management processes than Warehouse management II.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Inventory management, Sales and marketing, Procurement and sourcing</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics 365 for Operations version 1611.</td>
</tr>
</tbody>
</table>

**Worker reminders in Human Resources**

**Human Resources Payroll information**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Low usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Human resources</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics 365 for Operations version 1611</td>
</tr>
</tbody>
</table>

**Workflow for creating goals**

A workflow for managing the creation of employee goals is one of several workflows that were available to help coordinate the performance management process.

| Reason for deprecation/removal | Performance management has been completely redesigned in Finance and Operations. |
The redesigned Performance management feature gives more control over the content of the goals, the measurements that are used to track progress, and the attachment of supporting documentation. Goals can be stored as templates and then reused. This feature can help you set up additional goals for your employees more quickly.

| Replaced by another feature? | 
|-------------------------------|---|
| Product areas affected | Human capital management |
| Status | Removed as of Dynamics 365 for Operations version 1611. |

### Dynamics AX 7.0

#### Ability to cancel changes to a vendor invoice

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Performance enhancement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

#### AIF, AxD, and AxBC integrations

In Application Integration Framework (AIF), data can be exchanged with external systems through business logic that is exposed as services. Dynamics AX includes services that are based on documents and .NET Business Connector (AxBC). A document is created by using XML. The XML includes header information that is added to create a *message* that can be transferred into or out of Dynamics AX. Examples of documents include sales orders and purchase orders. However, almost any entity, such as a customer, can be represented by a document. Services that are based on documents use the *AxDocument* classes.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The architecture of AIF and AxDs could not be scaled to a cloud service. There were performance issues around bulk import.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>This feature is replaced by the Data Import/Export framework, which supports recurring bulk import/export. For AxBC, we recommend that you use the actual tables.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>AxDs, AxBCs, and AIF</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

#### Billing code rate scripts

Billing scripts were used to calculate billing rates for billing codes. This scripts required custom development in the C Sharp or Visual Basic programming language. In the current version of Dynamics AX, the **billing code rate scripts** are not supported.
<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The support for the custom C Sharp or Visual Basic scripts was not added in Dynamics AX 7.0.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Public sector, Accounts receivable</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**BOMs without BOM versions**

When the **BOM versions** configuration key was disabled, bill of materials (BOM) versions were hidden in all forms, and the system forced a 1:1 relationship between released products and BOMs. In the current version of Dynamics AX, the **BOM versions** configuration key can't be disabled.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Using a configuration key to control BOM versions doesn't scale in a cloud environment.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Product information management, Inventory management</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Brazilian Bordero**

Specific method of payment for Brazilian companies

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Support for the Brazilian Bordero method of payment has been discontinued from Brazilian localization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Brazilian Sintegra statement**

Federal tax statement for ICMS tax

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>This statement is no longer applicable in some Brazilian states.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No. Users can use Generic Electronic reporting tool to configure the statement if required under specific situations.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Fiscal books</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Brazilian SCAN contingency mode for NF-e**

(SCAN) contingency environment is used to generate, export, and import the status of a Nota Fiscal eletrônica (NF-e) when the environment of Secretaria da Fazenda (SEFAZ) is not available.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>This method of contingency is no longer applicable in all Brazilian states</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts receivable</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Business Analyzer**

This mobile application let users review key business metrics.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>This functionality has been replaced by another feature.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>The Monitor financial performance content pack for Microsoft Power BI will include key financial metrics that were previously available in Business Analyzer.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>General ledger</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: The use of Business Analyzer has been deprecated.</td>
</tr>
</tbody>
</table>

**Business statistics**

The setup of business statistics inquiries that can help you analyze the performance of the organization.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Legacy approach to business intelligence (BI), low customer usage, and a limited feature set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>New BI solutions for the current version of Dynamics AX</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Procurement and sourcing, Accounts payable, Sales and marketing, Accounts receivable</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
<tr>
<td>Change document date function in Invoice approval journal</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Reason for deprecation/removal</strong></td>
<td>Low usage</td>
</tr>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>Yes. The document date on the posted vendor transaction can be changed.</td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
<td>Accounts payable</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ClieOp03 payment format for the Netherlands</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reason for deprecation/removal</strong></td>
</tr>
<tr>
<td><strong>Replaced by another feature?</strong></td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
</tr>
<tr>
<td><strong>Status</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Compliance Center</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Compliance Center was an Enterprise Portal site for managing the documentation requirements for compliance initiatives that are related to the Sarbanes-Oxley law.</td>
</tr>
<tr>
<td><strong>Reason for deprecation/removal</strong></td>
</tr>
<tr>
<td><strong>Replaced by another feature?</strong></td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
</tr>
<tr>
<td><strong>Status</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connector for Microsoft Dynamics</th>
</tr>
</thead>
<tbody>
<tr>
<td>This tool was used to integrate key data from Microsoft Dynamics CRM to Dynamics ERP applications.</td>
</tr>
<tr>
<td><strong>Reason for deprecation/removal</strong></td>
</tr>
<tr>
<td><strong>Replaced by another feature?</strong></td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
</tr>
<tr>
<td>Status</td>
</tr>
<tr>
<td>---------------------------------</td>
</tr>
</tbody>
</table>

**Container unit and multi dimension on-hand**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Duplicate functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes. Since AX 2012, this functionality has been replaced by the consolidated batch orders feature set. This feature set includes the consolidated on-hand view.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Product information management, Production control, Inventory management, Sales and marketing</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Cue group metadata**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Cue groups were used to display one or more Cues in the FactBox area. There was limited uptake, and there were also performance concerns, because a record change in a parent form caused one query per Cue in the Cue group.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Cue metadata**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Cue metadata was limited to count or sum information.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Tile metadata was introduced to provide more flexibility for modeling. For example, you can model current counts, navigation, and key performance indicators (KPIs). Count tile metadata is the direct replacement of the Cue metadata.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Danish check format**
<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Support for the Danish check format layout has been discontinued, and the report has been removed from DK localization.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

### Data partitions

Data partitions provide a logical separation of data in the Dynamics AX database.

| Reason for deprecation/removal | Data partitions were introduced in Dynamics AX 2012 R2 to enable data isolation. In a common scenario, a company has subsidiaries, and the data from one subsidiary should not be visible to another subsidiary, even though both subsidiaries are managed by the same IT department. However, extra scripts and management overhead throughout the program were required in order to create new partitions and populate them with data, and to back up partition data. In the cloud, where we have access to platform as a service (PaaS) database services (Microsoft Azure SQL Database), it's much more efficient to use a database as the isolation container than to do isolation in the program. Regardless of whether data partitioning is required for subsidiaries, for multiple tenants, or just for scale, we believe that the scenarios can be handled better through multiple instances of Finance and Operations. |
| Replaced by another feature? | Customers using data partitions must use multiple instances of Finance and Operations if database level separation is a critical issue. |
| Product areas affected | All modules |
| Status | Removed as of Dynamics AX 7.0. |

### Database and file share storage for attachments

Dynamics AX 2012 allowed storage of attachments in the database and in file shares. Both of those options are no longer supported.
| Reason for deprecation/removal | Files share storage is no longer supported because cloud-hosted environments cannot communicate with local file shares. Database storage has been deprecated in favor of Azure Blob storage. Azure Blob storage is equivalent to storage in the database, as documents can only be accessed through Finance and Operations client forms. This provides the added benefit of providing storage that doesn't negatively affect the performance of the database. Blob storage is the default storage mechanism for Document Management and works immediately. |
| Replaced by another feature? | Database storage has been deprecated in favor of Azure Blob storage. |
| Product areas affected | All modules |
| Status | Removed as of Dynamics AX 7.0. |

**Delimitation**

| Reason for deprecation/removal | No use of the functionality was found. |
| Replaced by another feature? | No |
| Product areas affected | Time and attendance |
| Status | Removed as of Dynamics AX 7.0. |

**Desktop client**

| Reason for deprecation/removal | The Dynamics AX client experience has been redesigned to improve usability across multiple platforms and devices. |
| Replaced by another feature? | The new web client is based on the desktop Form metadata and programming model that have been modified to provide a rich web platform. |
| Product areas affected | All modules |
| Status | Removed as of Dynamics AX 7.0. |

**Direct database connection**

In Dynamics AX 2012 R3, Retail Modern POS could connect directly to the Channel DB in similar fashion to Enterprise POS. This was in addition to the standard communication method of Retail Modern POS communicating through Retail Server.
### Reason for deprecation/removal

Direct database connectivity required lower security protocols and was primarily used to achieve the highest levels of performance. Due to the performance and security enhancements that have occurred in Finance and Operations, this functionality now causes more issues than it solves.

### Replaced by another feature?

No. Only standard Retail Server communication is now supported.

### Product areas affected

Channel DB/Retail Modern POS

### Status

Removed as of Dynamics AX 7.0.

### Dutch SWIFT MT940

#### Reason for deprecation/removal

Generic functionality is now used instead of localized functionality.

#### Replaced by another feature?

Yes, this functionality has been replaced by Advanced bank reconciliation functionality.

#### Product areas affected

All modules

#### Status

Deprecated: A removal date has not been set for this feature.

### eBilanz (XBRL for Germany)

This functionality provided eXtensible Business Reporting Language (XBRL) output that is intended specifically for the German eBilanz taxonomy.

#### Reason for deprecation/removal

Lack of customer usage

#### Replaced by another feature?

This feature hasn't been replaced by another feature, but multiple specialized XBRL packages that provide rich XBRL functionality are available for the German market.

#### Product areas affected

Management Reporter

#### Status

Deprecated: A removal date has not been set for this feature.

### Enterprise Portal client

#### Reason for deprecation/removal

A single client platform has been provided.
<table>
<thead>
<tr>
<th>Replaced by another feature?</th>
<th>The new web client is based on the desktop form metadata and programming model that have been modified to provide a rich web platform.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product areas affected</td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Environmental sustainability**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Low customer usage and a limited feature set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Compliance and internal controls, Accounts payable</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Form ActiveX and Managed Host controls**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The ActiveX and Managed Host controls are based on the deprecated desktop client.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>The extensible control framework supports building new controls that are based on HTML, CSS, and JavaScript, and is a first-class control in the Microsoft Visual Studio Tooling environment.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Generate prenotes by using a batch**

Prenote generation can't be done by using a batch, but it can still be done by a user.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>No form exists to persist and display the resulting prenote file when it's generated by using a batch.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Prenotes can still be generated, and the user has control over the location where the file is saved.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable, Accounts receivable, Cash and bank management</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of AX 7.0.</td>
</tr>
</tbody>
</table>

**German DTAUS payment export and account statement import (totals and transactions)**
<table>
<thead>
<tr>
<th>Feature</th>
<th>Reason for deprecation/removal</th>
<th>Replaced by another feature?</th>
<th>Product areas affected</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>German DTAZV payment format in domestic Currency</td>
<td>The format is no longer applicable in Germany, because it has been replaced by SEPA functionality.</td>
<td>Yes, this functionality has been replaced by SEPA payment export and advanced bank reconciliation functionality for importing account statements.</td>
<td>All modules</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
<tr>
<td>German MT940 import</td>
<td>Generic functionality is now used instead of localized functionality.</td>
<td>Yes, this functionality has been replaced by Advanced bank reconciliation functionality.</td>
<td>Accounts payable</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
<tr>
<td>German XML EU Sales list</td>
<td>The XML format for German EU Sales List reporting is no longer supported. Only the ELMAS5 text file format can be used to submit the EU Sales List report to the German Tax Office.</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Tax</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>-----</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**GL SSRS reports**

Reports that include the following menu items have been removed: *Summary trial balance, Detailed trial balance, Chart of accounts, Audit trail, Balances, and Balance list.*

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Financial Microsoft SQL Server Reporting Services (SSRS) reports have been replaced by Management Reporter capabilities and default reports.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Management Reporter (labeled <em>Financial reporting</em> in the current version of Dynamics AX)</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>General ledger</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**InfoPart and FormPart metadata**

InfoPart and FormPart metadata enabled the creation of FactBoxes for two different clients.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>InfoPart metadata, which was a simplified form definition, is converted into a Form by upgrade tooling. FormPart metadata, which referenced a Form, is replaced by a more direct reference that is created by upgrade tooling.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Main accounts contains the same list of accounts that the Main account list page contained. The grid view in Main accounts also shows an even smaller, grid-like view.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Main account list page**

A list of accounts for the legal entity and related balance information

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Balance information is available on the Trial balance list page by account and dimension.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Main accounts contains the same list of accounts that the Main account list page contained. The grid view in Main accounts also shows an even smaller, grid-like view.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>General ledger</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

**Malaysia and Singapore bank cash flow report**

This feature let the user print a cash flow report that shows transactions and details of the cash inflows and outflows for a specific date range for selected bank accounts.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The same information can be obtained from the Inquiry bank transaction.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>The Inquiry bank transaction</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Cash and bank management</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Mexican CFD electronic invoice**

This feature enabled the generation of Mexican electronic invoices by using the Comprobante Fiscal Digital (CFD) method, where the company signs the invoice by requesting the related authorization from the government. This feature also provides a monthly report that includes all electronic invoices that were issued in the period.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The method is no longer applicable. The generation of electronic invoices by using the CFD method was deprecated by the tax authorities and replaced by the Comprobante Fiscal Digital a través de Internet (CFDI) method, where the signing is delegated to the third-party provider (PAC). The monthly report has been removed, and an inquiry option lets users inquire about historical transactions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Account receivables, Project</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Mexico realized and unrealized VAT**

Dynamics AX 2012 managed unrealized value-added tax (VAT) by using Mexico-specific functionality for unrealized tax.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Duplicate functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes, this functionality has been replaced by standard conditional sales tax functionality that is provided by Core.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Tax</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Deprecated: A removal date has not been set for this feature.</td>
</tr>
</tbody>
</table>

**Microsoft Outlook integration**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>This functionality has been replaced by Microsoft Exchange Server integration.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
<td>Sales and marketing</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Private blocking of inventory and warehouse management journals**

The inventory and warehouse journals no longer support the ability to mark a journal as private for a selected user. Only the process of blocking journals as private for user groups and blocking during editing is supported.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>No use of the functionality was found.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>No</td>
</tr>
<tr>
<td><strong>Product areas affected</strong></td>
<td>Inventory management</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Product builder**

Product builder was used to dynamically configure items from a sales order, purchase order, production order, sales quotation, project quotation, or item requirement. Based on a product model that had modeling variables, the user could select values to meet the customer requirements and get a unique product variant that had a BOM and route.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Product builder exposed X++ code to end users and isn't supported in the current version of Dynamics AX. It has been removed to avoid duplicate maintenance efforts on overlapping, sizeable codebases.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Replaced by another feature?</strong></td>
<td>Yes. The constraint-based configuration was introduced in Dynamics AX 2012 where the depreciation of Product builder in future versions was already announced. The constraint-based configuration technology is selected on the product masters to enable the configuration. To learn more, see Product configuration overview.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Product information management, Sales and marketing</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

### Production Floor app

This is the app for tablet devices running Windows 8.1 RT and Windows 8.1 Pro.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>With the change to a web-based client, it is possible to deliver similar functionality through the native Dynamics AX 7.0 client. The Job Card Device provides a production floor user interface that is optimized for touch and tablet form factors.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Yes. The Job Card Device, which is a native part of Dynamics AX 7.0.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Production control</td>
</tr>
<tr>
<td>Status</td>
<td>Deprecated: A removal date from the Microsoft store has not yet been set for this feature.</td>
</tr>
</tbody>
</table>

### Rename product dimension

This feature let you change the name of one of the three standard product dimensions (size, color, or style) to a name that better suited your business requirements. Renaming included all the labels where the product dimension name was used.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The current version of Dynamics AX doesn't support label changes at run time.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Product information management</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

### Retail Server connectivity using HTTP

In Dynamics AX 2012 R3, the Retail Server could function using HTTP communication (non-secured). This was in addition to the standard communication using HTTPS.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Due to new security requirements, only secured communication using TLS 1.2 (or above, as available) is now supported. The self-service installer will automatically configure the computer for this communication.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No. Only standard HTTPS communication is now supported.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Retail Server</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

### Role Center pages

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Role Center pages were built on the deprecated Enterprise Portal platform, which has been replaced by the new web client platform in the current version of Dynamics AX.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>The new Workspace form pattern provides users with a process-centered design that provides easy access to commonly used tasks within that process.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

### Sales tax jurisdictions

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Low customer usage and a limited feature set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>US sales tax</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

### Sites Services

Sites Services let you build websites that extend your business processes to the Internet without IT support.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The Microsoft Azure infrastructure that is used by Dynamics AX has new capabilities that can be used instead (for example, Azure sites).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>HR recruiting, Case management, Request for quotes, Vendor registration, Collaborative workspaces for opportunities and campaigns</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

### SSAS demand forecasting strategy
<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>The design of the feature cannot be supported in the new cloud architecture.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Azure Machine Learning demand forecasting strategy</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Master planning</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Vendor invoice pool excluding posting details**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Low usage. This functionality has been replaced by the Invoice journal that has workflow functionality.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Workflow capabilities of the Invoice journal.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Accounts payable</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
</tbody>
</table>

**Virtual company accounts**

The virtual companies feature is no longer supported in Dynamics AX. The virtual companies feature let users set up tables that could be shared by a set of companies. For a description of the feature, see [Company accounts](#) and [Virtual company accounts](#). The feature works by grouping tables into collections that are assigned to virtual companies, which are groups of existing “real” companies. Queries are created so that all the companies in the virtual company can access the data in the tables of the associated table collections.

| Reason for deprecation/removal | - Virtual companies must be set up before data is stored in the tables. Retrofitting virtual companies onto an existing implementation is very difficult.  
- Because there has been so much data normalization in the current version of Dynamics AX, it has become difficult to know what to add to the table collections. For example, it’s difficult to know which tables to share. All the tables referenced from tables that are in a virtual company must also added. Because of table normalization, even simple master data that is spread across multiple tables must be part of the virtual company. Any mistake that is made here will cause functional issues.  
- When a table is part of a virtual company, it loses information about the origin of the data, and only the virtual company is recorded.  
Global tables can be used to make tables accessible from all companies. Currently, there is no replacement. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Product areas affected</td>
<td>All modules</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 7.0.</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>--------------------------------</td>
</tr>
</tbody>
</table>

**Windows 8 tablet app**

The Windows 8 tablet app provided functionality for expense entry and approval.

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Finance and Operations is compatible with tablets. The tablet app is no longer required.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Expense management</td>
</tr>
<tr>
<td>Status</td>
<td>Removed: This functionality is only available for Dynamics AX 2012 R3.</td>
</tr>
</tbody>
</table>

**Workplanner**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>Low usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>No, but the Profile relation page, which is opened from the Profile groups page, supports the same business scenario as the deprecated Workplanner page.</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>Time and attendance</td>
</tr>
<tr>
<td>Status</td>
<td>The code has not been removed. However, the form, JmgWorkPlanner, was not migrated.</td>
</tr>
</tbody>
</table>

**X++ financial statements**

<table>
<thead>
<tr>
<th>Reason for deprecation/removal</th>
<th>This functionality has been replaced by another feature.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replaced by another feature?</td>
<td>Management Reporter (labeled Financial reporting in the current version of Dynamics AX)</td>
</tr>
<tr>
<td>Product areas affected</td>
<td>General ledger</td>
</tr>
<tr>
<td>Status</td>
<td>Removed as of Dynamics AX 2012</td>
</tr>
</tbody>
</table>
As the Microsoft code base continues to evolve, some methods and metadata elements will no longer be required. Microsoft will mark these obsolete methods and metadata elements for deprecation.

- Methods are marked with the `SysObsolete` attribute. Typically, this attribute recommends an alternative to the method.
- For metadata elements, the `IsObsolete` property is set to `Yes`.

The deprecation is compatible with both binaries and design time. The referencing code will continue to work as expected, and no immediate action is required. During compilation, any references to deprecated artifacts are reported as compile **warnings**.

### Cleanup of deprecated elements

After a period of at least 12 months, Microsoft might delete obsolete methods and metadata elements.

However, if telemetry shows that any obsolete methods or metadata elements are still used, Microsoft will **not** delete them, to reduce the risk that consumers will be broken.

### Minimize your risk of being affected

Here are some tips that you, as a consumer of the Microsoft code base, can use to avoid being affected when methods and metadata elements are deprecated:

- Compile your code base at least every 12 months on top of the latest code base. If you receive any warnings because deprecated artifacts are used, address those warnings as soon as possible.
- Avoid **new** dependencies on deprecated artifacts. Microsoft might have just deleted the artifact, because there is a time window between when releases and telemetry are available.

### List of deprecated methods and metadata elements

For reference, download the Microsoft Excel file, `ObsoleteElementsPerVersion.xlsx`, which shows the artifacts that have been marked for deprecation in each major release.
This document provides the list of deprecated APIs and migration guidance for some of the deprecated APIs.

Overview

A number of APIs from Dynamics AX 2012 have been identified. The reason for the deprecation for each API varies. Most commonly, the reasons are one of the following:

- Not suited/applicable to the new client.
- Degrade performance.
- Chatty (cause lot of traffic back and forth between server and client).
- Redundant (framework automatically handles these now).

Throughout this table, under the **Reason for Deprecation** heading, "the client" refers to the web client.

List of deprecated APIs

<table>
<thead>
<tr>
<th>OBJECT</th>
<th>TYPE</th>
<th>NAME</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActionPane</td>
<td>Method</td>
<td>tabChanged</td>
<td>Updates to ActionPanes (or controls inside of ActionPanes) should be done based on the active row, not when the tab becomes active.</td>
</tr>
<tr>
<td>ActionPaneTab</td>
<td>Method</td>
<td>selectionChanged</td>
<td>Updates to ActionPaneTabs (or controls inside of ActionPaneTabs) should be done based on the active row, not when the tab becomes active.</td>
</tr>
<tr>
<td>Box</td>
<td>Method</td>
<td>yesNoTextMenu-LinkText</td>
<td>N/A</td>
</tr>
</tbody>
</table>
| ComboBox      | Method  | getEditText        | Overview
Reason for deprecation
Redundant.
Migration notes
Use getText instead. |
<table>
<thead>
<tr>
<th>OBJECT</th>
<th>TYPE</th>
<th>NAME</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataSet</td>
<td>Class</td>
<td>DataSetNode</td>
<td>Used in Dynamics AX 2012 with Enterprise Portal.</td>
</tr>
<tr>
<td>DataSetRun</td>
<td>Class</td>
<td>DataSetRun</td>
<td>Reason for deprecation: Not applicable in the client.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Migration notes: Remove calls to these APIs from your code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DataSourceMethodInfo</td>
<td>Reason for deprecation: Specific to Dynamics AX 2012 Windows client and not compatible with the client.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DataSourceMethodInfoList</td>
<td>Migration notes: Remove usage of these APIs from your code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DDEClient DDEServer DLL DDEServer DLL Function HDC HWND Thread WinAPI Native WinGDI</td>
<td>Reason for deprecation: N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Migration notes: Remove calls to these APIs from your code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Form</td>
<td>Reason for deprecation: Used in Dynamics AX 2012 with address bar.</td>
</tr>
<tr>
<td></td>
<td>Method</td>
<td>addhistory</td>
<td>Migration notes: Remove calls to these APIs from your code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>currentHistoryName</td>
<td>Reason for deprecation: This is not a clean and recommended way to register override methods.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>currentHistoryState</td>
<td>Migration notes: Use registerOverrideMethod instead.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>updateHistory</td>
<td>Reason for deprecation: Used in Dynamics AX 2012 to register override methods.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Migration notes: Use registerOverrideMethod instead.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>arrange</td>
<td>Reason for deprecation: Used in Dynamics AX 2012 to register override methods.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>controlCallingMethod</td>
<td>Migration notes: Use registerOverrideMethod instead.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>controlMethod-Overload</td>
<td>Reason for deprecation: Used in Dynamics AX 2012 to register override methods.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>controlMethod-OverloadObject</td>
<td>Migration notes: Use registerOverrideMethod instead.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>copy cut paste</td>
<td>Reason for deprecation: Used in Dynamics AX 2012 to register override methods.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Migration notes: Use registerOverrideMethod instead.</td>
</tr>
<tr>
<td>OBJECT</td>
<td>TYPE</td>
<td>NAME</td>
<td>NOTES</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>----------------------------------</td>
<td>-------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Form</td>
<td>Method</td>
<td>delAutoCompleteString</td>
<td>Overview Used in Dynamics AX 2012 to set, get, and delete automatic suggestions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>getAutoCompleteString</td>
<td>Reason for deprecation Specific to Dynamics AX 2012 Windows client.</td>
</tr>
<tr>
<td>Form</td>
<td></td>
<td>setAutoCompleteString</td>
<td>Migration notes Remove calls to these APIs from your code.</td>
</tr>
<tr>
<td>Form</td>
<td>Method</td>
<td>autoCompleteString</td>
<td>Overview Used in Dynamics AX 2012 to set, get, and delete automatic suggestions.</td>
</tr>
<tr>
<td>Form</td>
<td>Method</td>
<td>firstField</td>
<td>Reason for deprecation Specific to Dynamics AX 2012 Windows client.</td>
</tr>
<tr>
<td>Form</td>
<td></td>
<td>formOnTop</td>
<td>Migration notes Remove calls to these APIs from your code.</td>
</tr>
<tr>
<td>Form</td>
<td>Method</td>
<td>hWnd installMessageProc</td>
<td>Reason for deprecation Specific to Dynamics AX 2012 Windows client and not compatible with the client.</td>
</tr>
<tr>
<td>Form</td>
<td></td>
<td>removeMessageProc</td>
<td>Migration notes Remove usage of these APIs from your code.</td>
</tr>
<tr>
<td>Form</td>
<td>Method</td>
<td>isPreloadedInstance</td>
<td>Reason for deprecation Preloading is not applicable in the client.</td>
</tr>
<tr>
<td>Form</td>
<td></td>
<td>lastField nextField nextGroup</td>
<td>Migration notes Remove calls to these APIs from your code.</td>
</tr>
<tr>
<td>Form</td>
<td>Method</td>
<td>prevField prevGroup</td>
<td>Reason for deprecation Preloading is not applicable in the client.</td>
</tr>
<tr>
<td>Form</td>
<td></td>
<td>nextGroup</td>
<td>Migration notes Remove calls to these APIs from your code.</td>
</tr>
<tr>
<td>OBJECT</td>
<td>TYPE</td>
<td>NAME</td>
<td>NOTES</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Form</td>
<td>Method</td>
<td>Lock</td>
<td><strong>Overview</strong> These methods were used to prevent the redrawing of windows when performing a set of UI updates. Without these the window would be redrawn in response to each individual change leading to bad end-user experience and degraded performance. <strong>Reason for deprecation</strong> These methods are specific to the Windows client and are no longer needed for the client. <strong>Migration notes</strong> A code upgrade rule has been provided to remove occurrences of these APIs. You can safely remove any calls to these APIs from your code.</td>
</tr>
<tr>
<td>Form</td>
<td>Method</td>
<td>lockWindowUpdate</td>
<td></td>
</tr>
<tr>
<td>Form</td>
<td>Method</td>
<td>unLock</td>
<td></td>
</tr>
<tr>
<td>Form</td>
<td>Method</td>
<td>print printPreview send</td>
<td><strong>Overview</strong> Used in Dynamics AX 2012 to override the Auto Report generation for the form <strong>Reason for deprecation</strong> Microsoft 365 integration offers a better user experience in the client. The ‘Export’ function is available for the user in the Dynamics AX client forms. <strong>Migration notes</strong> Remove calls to these APIs from your code.</td>
</tr>
<tr>
<td>Form</td>
<td>Method</td>
<td>redraw</td>
<td></td>
</tr>
<tr>
<td>Form</td>
<td>Method</td>
<td>resetStatusBarBackgroundColor</td>
<td></td>
</tr>
<tr>
<td>Form</td>
<td>Method</td>
<td>setStatusBarBackgroundColor</td>
<td></td>
</tr>
<tr>
<td>Form</td>
<td>Method</td>
<td>sysColorChanged</td>
<td></td>
</tr>
<tr>
<td>Form</td>
<td>Method</td>
<td>reload</td>
<td><strong>Overview</strong> Used to control styles or colors. <strong>Reason for deprecation</strong> Remove ability for developers to specify the colors via API for consistent visuals. <strong>Migration notes</strong> A code upgrade rule has been provided to remove occurrences of the redraw API. Remove usage of these APIs from your code.</td>
</tr>
<tr>
<td>OBJECT</td>
<td>TYPE</td>
<td>NAME</td>
<td>NOTES</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------</td>
<td>-----------------</td>
<td>--------------------------------------------</td>
</tr>
</tbody>
</table>
| Form                        | Method     | resize          | **Overview**  
This method was used when controls were added/removed from a form causing its size to change. Without it the window might not be correctly sized to account for the added/removed controls.  
**Reason for deprecation**  
These methods are specific to the Windows client and are no longer needed for the client.  
**Migration notes**  
You can safely remove any calls to these APIs from your code. |
| FormActiveXControl          | Class      |                 | **Overview**  
These were used to host or create various custom controls for Dynamics AX 2012.  
**Reason for deprecation**  
These technologies will not work with the client.  
**Migration notes**  
Application developers need to build replacement controls where needed using the control extensibility features. |
| FormAnimateControl          |            |                 |                                            |
| FormBuildActiveXControl     |            |                 |                                            |
| FormBuildAnimateControl     |            |                 |                                            |
| FormBuildManaged-HostControl|            |                 |                                            |
| FormBuildSegmented-EntryControl|        |                 |                                            |
| FormManagedHostControl      |            |                 |                                            |
| FormSegmented-EntryControl  |            |                 |                                            |
| FormControl                 | Method     |                 | **Overview**  
Used to enable drag-and-drop scenarios in Dynamics AX 2012.  
**Reason for deprecation**  
Drag-and-drop scenarios are not supported in the client.  
**Migration notes**  
Remove usage of these APIs from your code and refactor to enable the scenarios without dependency on drag-and-drop functionality. |
<p>| FormControl                 | Method     | beginDrag       |                                            |
|                            |            | dragDrop        |                                            |
|                            |            | dragLeave       |                                            |
|                            |            | dragOver        |                                            |
|                            |            | dragOverEx      |                                            |
|                            |            | dragText        |                                            |
|                            |            | drop            |                                            |
|                            |            | dropEx          |                                            |
|                            |            | dropEx          |                                            |
|                            |            | dropFile        |                                            |
|                            |            | endDrag         |                                            |
| FormControl                 | Method     | calcControlSize |                                            |</p>
<table>
<thead>
<tr>
<th>OBJECT</th>
<th>TYPE</th>
<th>NAME</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>FormControl</td>
<td>Method</td>
<td>command processBase</td>
<td>Overview: Was marked for deprecation in Dynamics AX 2012. Reason for deprecation: N/A. Migration notes: Remove calls to these APIs from your code.</td>
</tr>
<tr>
<td>FormControl</td>
<td>Method</td>
<td>processForm processLink</td>
<td></td>
</tr>
<tr>
<td>FormControl</td>
<td>Method</td>
<td>processPicture processTitle</td>
<td></td>
</tr>
<tr>
<td>FormControl</td>
<td>Method</td>
<td>context showContextMenu</td>
<td>Overview: This method was used when controls were added/removed from a form causing its size to change. Without it the window might not be correctly sized to account for the added/removed controls. Reason for deprecation: These methods relied on APIs that are specific to the Windows client. Migration notes: Use getMenuOptions and selectedMenuOptions instead.</td>
</tr>
<tr>
<td>FormControl</td>
<td>Method</td>
<td>copy cut paste</td>
<td></td>
</tr>
<tr>
<td>FormControl</td>
<td>Method</td>
<td>dateTextChange</td>
<td></td>
</tr>
<tr>
<td>FormControl</td>
<td>Method</td>
<td>editControl</td>
<td></td>
</tr>
<tr>
<td>FormControl</td>
<td>Method</td>
<td>hasControl-PositionOverride</td>
<td></td>
</tr>
<tr>
<td>FormControl</td>
<td>Method</td>
<td>helpField</td>
<td></td>
</tr>
<tr>
<td>FormControl</td>
<td>Method</td>
<td>hWnd</td>
<td>Overview: N/A. Reason for deprecation: Specific to Dynamics AX 2012 Windows client and not compatible with the client. Migration notes: Remove usage of these APIs from your code.</td>
</tr>
<tr>
<td>FormControl</td>
<td>Method</td>
<td>inputSearch</td>
<td></td>
</tr>
<tr>
<td>FormControl</td>
<td>Method</td>
<td>itemChanging</td>
<td></td>
</tr>
<tr>
<td>FormControl</td>
<td>Method</td>
<td>keyDown</td>
<td></td>
</tr>
<tr>
<td>OBJECT</td>
<td>TYPE</td>
<td>NAME</td>
<td>NOTES</td>
</tr>
<tr>
<td>-------------</td>
<td>--------</td>
<td>-------------------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>FormControl</td>
<td>Method</td>
<td>labelMouseDblClick mouseDblClick</td>
<td>Overview</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The FormControl.labelMouseDblClick (int x, int y, int button, Boolean Ctrl, Boolean Shift) method is called when the label for a control is double-clicked. It provides the x, y co-ordinates of the mouse pointer, a Boolean to indicate which mouse button was clicked and Booleans to indicate whether the Ctrl and Shift key were pressed. The FormControl.mouseDblClick (int x, int y, int button, Boolean Ctrl, Boolean Shift) method is similar in function to the labelMouseDblClick method. The difference is that this method is called whenever there is a double-click (not just on the labels). Reason for deprecation The double-click action does not translate well to web-based application and touch-based scenarios. Additionally they might end up being chatty in many instances. Migration notes The recommended replacement for these methods is to use a button and the clicked event.</td>
</tr>
<tr>
<td>FormControl</td>
<td>Method</td>
<td>mousedown mouseDblClick mouseDown mouseUp mouseEnter mouseLeave mouseMove mouseUp</td>
<td>Overview</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Used to detect and respond to mouse events. Reason for deprecation These are not touchscreen friendly and not supported in the client. Migration notes Remove usage of these APIs from your code and refactor to enable the scenarios without dependency on mouse events.</td>
</tr>
<tr>
<td>FormControl</td>
<td>Method</td>
<td>onHScroll onVScroll</td>
<td></td>
</tr>
<tr>
<td>FormControl</td>
<td>Method</td>
<td>paint</td>
<td></td>
</tr>
<tr>
<td>OBJECT</td>
<td>TYPE</td>
<td>NAME</td>
<td>NOTES</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------</td>
<td>-----------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| FormControl             | Method    | prefColumnSize              | **Overview**  
Used in Dynamics AX 2012 to control width and height  
**Reason for deprecation**  
Not applicable in the client.  
**Migration notes**  
Set the width and height explicitly instead. |
| FormControl             | Method    | selectionChanging           |                                                                                                                                          |
| FormControl             | Method    | setScrollInfo               |                                                                                                                                          |
| FormControl             | Method    | size                        |                                                                                                                                          |
| FormControl             | Method    | updateWindow                |                                                                                                                                          |
| FormControl / FormDesign| Property  | AcquireFocus                |                                                                                                                                          |
| FormControl / FormDesign| Property  | ActiveBackCol               | **Overview**  
Used to control styles or colors.  
**Reason for deprecation**  
Remove ability for developers to specify the colors via API for consistent visuals.  
**Migration notes**  
Remove usage of these APIs from your code. |
|                        |           | ActiveBackColor             |                                                                                                                                          |
|                        |           | ActiveBackColorRGB          |                                                                                                                                          |
|                        |           | ActiveForeColor             |                                                                                                                                          |
|                        |           | ActiveForeColorRGB          |                                                                                                                                          |
|                        |           | AlternateRowShading         |                                                                                                                                          |
|                        |           | BackgroundColor             |                                                                                                                                          |
|                        |           | BackgroundColorRGB          |                                                                                                                                          |
|                        |           | BackStyle                  |                                                                                                                                          |
|                        |           | BackStyleRGB               |                                                                                                                                          |
|                        |           | CharacterSet                |                                                                                                                                          |
|                        |           | ColorScheme                 |                                                                                                                                          |
|                        |           | DrawFocusRect               |                                                                                                                                          |
|                        |           | ForegroundColor             |                                                                                                                                          |
|                        |           | ForegroundColorRGB          |                                                                                                                                          |
|                        |           | GridLines                  |                                                                                                                                          |
|                        |           | GridLinesStyle             |                                                                                                                                          |
|                        |           | PromptRect                  |                                                                                                                                          |
| FormControl / FormDesign| Property  | AlignChild                  | **Overview**  
Used to control layout.  
**Reason for deprecation**  
Remove ability for developers to control layout using this property to achieve a consistent layout.  
**Migration notes**  
Remove usage of these APIs from your code. Use styles or CSS instead. |
<p>|                        |           | AlignChildren               |                                                                                                                                          |
|                        |           | Border                      |                                                                                                                                          |
|                        |           | BottomMargin                |                                                                                                                                          |
|                        |           | BottomMarginMode            |                                                                                                                                          |
|                        |           | ColumnSpace                 |                                                                                                                                          |
|                        |           | ColumnSpaceMode             |                                                                                                                                          |
|                        |           | ColumnSpaceValue            |                                                                                                                                          |
|                        |           | LeftMargin                  |                                                                                                                                          |
|                        |           | LeftMarginMode              |                                                                                                                                          |
|                        |           | RightMargin                 |                                                                                                                                          |
|                        |           | RightMarginMode             |                                                                                                                                          |
|                        |           | SizeHeight                  |                                                                                                                                          |
|                        |           | SizeWidth                   |                                                                                                                                          |
|                        |           | TabAppearance               |                                                                                                                                          |
|                        |           | TabAutoChange               |                                                                                                                                          |
|                        |           | TabLayout                   |                                                                                                                                          |
|                        |           | TabMode                     |                                                                                                                                          |
|                        |           | TabPlacement                |                                                                                                                                          |
|                        |           | Top                          |                                                                                                                                          |
|                        |           | TopMargin                   |                                                                                                                                          |
|                        |           | TopMarginMode               |                                                                                                                                          |
|                        |           | TopMode                     |                                                                                                                                          |
|                        |           | VerticalSpacing             |                                                                                                                                          |
|                        |           | VerticalSpacingMode         |                                                                                                                                          |
|                        |           | VerticalSpacingValue        |                                                                                                                                          |</p>
<table>
<thead>
<tr>
<th>OBJECT / Property</th>
<th>Type</th>
<th>NAME</th>
<th>NOTES</th>
</tr>
</thead>
</table>
| FormControl / FormDesign  | Property | AllowDocking AlwaysOnTop ArrangeGuide ArrangeWhen ContainerScroll- HorizontalOffset ContainerScroll- VerticalOffset IMEMode MaximizeBox MinimizeBox Mode NeededAccessLevel ProgressType Securable SecurityKey StatusBarStyle WindowResize | **Overview**
N/A
**Reason for deprecation**
Specific to Dynamics AX 2012 Windows client, no longer needed.
**Migration notes**
Remove usage of these APIs from your code. |
| FormControl / FormDesign  | Property | Bold                                                                  |                                                |
| FormControl / FormDesign  | Property | CanScroll                                                             |                                                |
| FormControl / FormDesign  | Property | DisabledImage DisabledImageLocation DisabledResource                  |                                                |
| FormControl / FormDesign  | Property | DisplayTarget HyperLinkDataSource HyperLinkMenuItem SaveFilter SaveSize | **Overview**
Used in Dynamics AX 2012 with Enterprise Portal
**Reason for deprecation**
Not applicable in the client.
**Migration notes**
Remove calls to these APIs from your code. |
| FormControl / FormDesign  | Property | Font                                                                  |                                                |
| FormControl / FormDesign  | Property | FontSize                                                              |                                                |
| FormControl / FormDesign  | Property | Frame FramePosition                                                  | **Overview**
N/A
**Reason for deprecation**
Remove ability for developers to control frames via metadata.
**Migration notes**
Remove usage of these APIs from your code. |
| FormControl / FormDesign  | Property | HideToolbar HorizontalScrollBarVisible Scrollbars VerticalScrollBarVisible | **Overview**
N/A.
**Reason for deprecation**
Remove ability for developers to control scrollbars via metadata.
**Migration notes**
Remove usage of these APIs from your code. |
<p>| FormControl / FormDesign  | Property | ImageMode                                                             |                                                |
| FormControl / FormDesign  | Property | ImageName                                                             |                                                |</p>
<table>
<thead>
<tr>
<th>OBJECT</th>
<th>TYPE</th>
<th>NAME</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>ImageResource</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>Italic</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>LabelAlignment</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>LabelBold</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>LabelCharacterSet</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>LabelFont</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>LabelFontSize</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>LabelForegroundColor</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>LabelForegroundColorRGB</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>LabelGuide</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>LabelHeight</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>LabelHeightMode</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>LabelHeightValue</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>LabelItalic</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>LabelUnderline</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>LabelWidth</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>LabelWidthMode</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>LabelWidthValue</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>Location</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>NormalResource</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>ParentPage</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>SearchAfterInput</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>SearchMode</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>SelectControl</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>SendExternalContext</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>ShortKey</td>
<td></td>
</tr>
<tr>
<td>FormControl / FormDesign</td>
<td>Property</td>
<td>Underline</td>
<td></td>
</tr>
<tr>
<td>FormDataRow</td>
<td>Class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OBJECT</td>
<td>TYPE</td>
<td>NAME</td>
<td>NOTES</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------</td>
<td>--------------</td>
<td>---------------------------------------------------------</td>
</tr>
<tr>
<td>FormDataSource</td>
<td>Property</td>
<td>autoNotify</td>
<td>Overview: Was marked for deprecation in Dynamics AX 2012. Reason for deprecation: N/A. Migration notes: Remove usage from your code.</td>
</tr>
<tr>
<td>FormDataSource</td>
<td>Method</td>
<td>cacheOnlyMode</td>
<td></td>
</tr>
<tr>
<td>FormDataSource</td>
<td>Method</td>
<td>cacheRemoveRecord</td>
<td></td>
</tr>
<tr>
<td>FormDataSource</td>
<td>Method</td>
<td>defaultMark</td>
<td></td>
</tr>
<tr>
<td>OBJECT</td>
<td>TYPE</td>
<td>NAME</td>
<td>NOTES</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------</td>
<td>---------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>FormDataSource</td>
<td>Method</td>
<td>findRecord, findValue</td>
<td><strong>Usage</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The FormDataSource.findRecord( Common record) method finds a specific record in the data source and makes it the current record. The FormDataSource.findValue(F ieldId f ield, str v alue) method find a specific value in a specific field in the data source and makes the corresponding record the current record. It uses the FormDataSource.findRecord method for this. <strong>Reason for deprecation</strong> These methods use linear searching and load a large number of records in memory and negatively impact performance. <strong>Migration notes</strong> Replace with new APIs. Replace findRecord with positionToRecord and findValue with positionToRecordByValue. New APIs do not work in some cases, most notably with Temp tables and Views. The framework will throw an exception in those cases. If replacing with new APIs is not possible, recommended replacement is to call element.args(). lookupRecord(recordToFind) Followed by FormDataSource.research(false): FormDataSource is the data source that contains the record you want to find. Passing in a &quot;false&quot; argument to research causes it to not retain the current position since we want to change the position to the record we found using args.lookupRecord avoids resetting sort order, ranges, etc.</td>
</tr>
<tr>
<td>FormDataSource</td>
<td>Method</td>
<td>getDataRow</td>
<td></td>
</tr>
<tr>
<td>FormDataSource</td>
<td>Method</td>
<td>markAllLoadedRecords</td>
<td></td>
</tr>
<tr>
<td>OBJECT</td>
<td>TYPE</td>
<td>NAME</td>
<td>NOTES</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------</td>
<td>-------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| FormDataSource         | Method  | maxPagingRowCountValue                    | **Overview** Used in Dynamics AX 2012 with Enterprise Portal  
**Reason for deprecation** Not applicable in the client.  
**Migration notes** Remove calls to these APIs from your code. |
|                        |         | pagingEnabled                             |                                                                                                                                       |
|                        |         | startRowIndex                             |                                                                                                                                       |
|                        |         | setPagingParameters                      |                                                                                                                                       |
|                        |         | totalNumberOfRows                         |                                                                                                                                       |
| FormDataSource         | Method  | print                                     | **Overview** Used in Dynamics AX 2012 with Enterprise Portal  
**Reason for deprecation** Not applicable in the client.  
**Migration notes** Remove calls to these APIs from your code. |
| FormDesign             | Method  | cssClass localWebMenu                    | **Overview** Used in Dynamics AX 2012 with Enterprise Portal  
**Reason for deprecation** Not applicable in the client.  
**Migration notes** Remove calls to these APIs from your code. |
|                        |         | showWebHelp                               |                                                                                                                                       |
|                        |         | supportReload                             |                                                                                                                                       |
| FormObjectSetNotify    | Method  | onPagingParametersChanged                 | **Overview** Used in Dynamics AX 2012 with Enterprise Portal  
**Reason for deprecation** Not applicable in the client.  
**Migration notes** Remove calls to these APIs from your code. |
| FormObjectSetPaging.ParamsChangedEvtArgs | Class   |                                           |                                                                                                                                       |
| Global xInfo           | Method  | endLengthyOperation                      | **Overview** These methods were used to show/stop showing a progress indicator during long running operations.  
**Reason for deprecation** In the client, the system automatically takes care of showing/hiding the progress indicator and calls to these APIs are not needed.  
**Migration notes** You can safely remove any calls to these APIs from your code.  |
<p>|                        |         | startLengthyOperation                     |                                                                                                                                       |
| Image                  | Method  | captureScreen                             |                                                                                                                                       |
|                        |         | captureWindow                             |                                                                                                                                       |
|                        |         | clipboardCopy                             |                                                                                                                                       |
|                        |         | clipboardPaste                            |                                                                                                                                       |
|                        |         | crop                                      |                                                                                                                                       |
|                        |         | displayImage                              |                                                                                                                                       |
|                        |         | displayOrigin                             |                                                                                                                                       |
|                        |         | exportBitmap                              |                                                                                                                                       |
|                        |         | flip                                      |                                                                                                                                       |
|                        |         | getImageDimensionUnits                   |                                                                                                                                       |
|                        |         | getPixel                                  |                                                                                                                                       |
|                        |         | height                                    |                                                                                                                                       |
|                        |         | imageInfo                                 |                                                                                                                                       |
|                        |         | imageSpotlight                            |                                                                                                                                       |
|                        |         | promoteColor                              |                                                                                                                                       |
|                        |         | reduceColorOctree                         |                                                                                                                                       |
|                        |         | resize                                    |                                                                                                                                       |
|                        |         | rotate                                    |                                                                                                                                       |
|                        |         | saveImage                                 |                                                                                                                                       |
|                        |         | saveType                                  |                                                                                                                                       |
|                        |         | transparent                               |                                                                                                                                       |
|                        |         | width                                     |                                                                                                                                       |</p>
<table>
<thead>
<tr>
<th>OBJECT</th>
<th>TYPE</th>
<th>NAME</th>
<th>NOTES</th>
</tr>
</thead>
</table>
| ListPage Page     | Method | activeActionPane-TabNames   | Overview  
In the client, Action Pane tabs are handled client-side only, the server is not aware of the state.  
Reason for deprecation  
Migration notes Remove usage of this API from your code. |
| MessageWin        | Class  |                             |                                                                      |
| Object            | Method | notify notifyAll wait       | Overview  
Reason for deprecation  
Migration notes Calls to these APIs from formRun or its derivatives are allowed. Calls to these APIs from any other object should be removed. |
| Object            | Method | objectOnServer              | Overview  
Reason for deprecation  
Migration notes You can safely remove calls to these APIs from your code. It will always evaluate to true. |
<table>
<thead>
<tr>
<th>OBJECT</th>
<th>TYPE</th>
<th>NAME</th>
<th>NOTES</th>
</tr>
</thead>
</table>
| Object | Method| setTimeOut | **Overview** This method existed on Object, but was non-functional. The implementation on FormRun was used as a timer to delay the execution of a piece of logic.  
**Reason for deprecation** The browser based client no longer supported this implementation.  
**Migration notes** Use the new setTimeoutEx method on the FormRun instead. Note that the setTimeoutEx method expects the callback to accept a parameter of type AsyncTaskResult, example: myCallBack(AsyncTaskResult result). |
| PopupMenu | Class |        | **Overview** Used in Dynamics AX 2012 to get splitters that let users change the size of the two parts that are split.  
**Reason for deprecation** Relied on APIs that are specific to the Dynamics AX 2012 Windows Client and cannot be used with the client.  
**Migration notes** Use ContextMenu instead. |
| SysExcel | Class |        | **Overview** The SysExcel classes used COM to create and edit Excel workbooks.  
**Reason for deprecation** SysExcel relied on calls to Excel COM objects from the client. Those COM objects are not on the server and COM calls are highly discouraged going forward.  
**Migration notes** Use the OpenXML .NET framework APIs instead. We are investigating the creation of an assembly that wraps OpenXML to make it easier to call from X++. |
<table>
<thead>
<tr>
<th>OBJECT</th>
<th>TYPE</th>
<th>NAME</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysINetMail</td>
<td>Class</td>
<td></td>
<td><strong>Overview</strong> These email related classes used predominantly client-side technologies that are no longer available and/or are highly discouraged.</td>
</tr>
<tr>
<td>SysMailer</td>
<td>Class</td>
<td></td>
<td><strong>Reason for deprecation</strong> The SysINetMail class is being deprecated because it used client-side MAPI. The SysMailer class is being deprecated because it used CDO (a variant of OLE messaging). The classes beginning with SmmOutlook are being deprecated since they use Outlook COM objects.</td>
</tr>
<tr>
<td>SmmOutlook</td>
<td>Class</td>
<td></td>
<td><strong>Migration notes</strong> Sending email via SMTP using the SysMailerNet class will be supported going forward. We are also actively working on client-side interactive email capabilities.</td>
</tr>
<tr>
<td>SysFormSplitter</td>
<td>Class</td>
<td></td>
<td><strong>Overview</strong> Used in Dynamics AX 2012 to get splitters that let users change the size of the two parts that are split.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Reason for deprecation</strong> No longer needed in the client.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Migration notes</strong> Controls automatically provide the functionality. You can safely remove any calls to these APIs from your code. A code upgrade rule may be created in the future to automatically remove the usage.</td>
</tr>
<tr>
<td>SysListPageHelper</td>
<td>Class</td>
<td></td>
<td><strong>Overview</strong> Used to by classes to indirectly extend FormRun.</td>
</tr>
<tr>
<td>SysSetupFormRun</td>
<td>Class</td>
<td></td>
<td><strong>Reason for deprecation</strong> Has been merged with FormRun class.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Migration notes</strong> Use the FormRun class instead.</td>
</tr>
<tr>
<td>TextBuffer</td>
<td>Method</td>
<td>fromFile</td>
<td>Use the .NET StreamReader class instead.</td>
</tr>
<tr>
<td>OBJECT</td>
<td>TYPE</td>
<td>NAME</td>
<td>NOTES</td>
</tr>
<tr>
<td>----------------</td>
<td>---------</td>
<td>-----------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>TextBuffer</td>
<td>Method</td>
<td>toFile</td>
<td>Use the .NET StreamWriter class instead.</td>
</tr>
<tr>
<td>Thread</td>
<td>Class</td>
<td></td>
<td><strong>Overview</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Reason for deprecation</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specific to Dynamics AX 2012 Windows client and not compatible with the client.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Migration notes</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Consider replacing with the new runAsync method or remove usage of these APIs from your code.</td>
</tr>
<tr>
<td>WinAPI</td>
<td>Class</td>
<td></td>
<td><strong>Overview</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Reason for deprecation</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specific to Dynamics AX 2012 Windows client and not compatible with the client.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Migration notes</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Remove usage of these APIs from your code. Replace file access APIs, such as WinAPI::getTempPath, WinAPI::fileExists, with the new file APIs.</td>
</tr>
<tr>
<td>WinAPIServer</td>
<td>Method</td>
<td>cryptProtectData</td>
<td><strong>Overview</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>cryptUnprotectData</td>
<td>The WinAPI::cryptProtectData(CryptoBlob _unEncryptedDataBlob) and WinAPI::cryptUnprotectData(CryptoBlob _encryptedDataBlob) methods were used to encrypt and decrypt sensitive data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Reason for deprecation</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>These methods are best suited to desktop usage and not recommended for web-based application usage. They also have a negative impact on performance.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Migration notes</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Use the .NET framework APIs and well-known hashing/security algorithms instead.</td>
</tr>
<tr>
<td>OBJECT</td>
<td>TYPE</td>
<td>NAME</td>
<td>NOTES</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td>------------------</td>
<td>-------</td>
</tr>
<tr>
<td>xApplication</td>
<td>Method</td>
<td>runAsync</td>
<td>Overview: In Dynamics AX 2012 the <code>xApplication::runAsync</code> method was used to make asynchronous calls to methods. Reason for deprecation: Replaced with methods better suited to the client. Migration notes: Use <code>runAsync</code> methods on the <code>Global</code> or <code>FormRun</code> classes instead. These new versions of <code>runAsync</code> enable the caller to make an async call to a static X++ class method. They leverage the .NET <code>System.Threading.Tasks</code> library to execute an async method in X++. The use of the <code>System.Threading.Tasks.Task</code> type allows the developer to take advantage of the rich set of features available in .NET.</td>
</tr>
<tr>
<td>xGlobal</td>
<td>Method</td>
<td>clientKind</td>
<td>Overview: Most commonly used to detect presence of client, such as an interactive session. Reason for deprecation: Replaced with a method better suited to the client. Migration notes: Use <code>global::hasGUI</code> method instead.</td>
</tr>
<tr>
<td>xGlobal</td>
<td>Method</td>
<td>computerName</td>
<td></td>
</tr>
<tr>
<td>xGlobal</td>
<td>Method</td>
<td>forceFormPreload</td>
<td>Overview: Used in Dynamics AX 2012 with preloading. Reason for deprecation: Preloading is not applicable in the client. Migration notes: Remove calls to these APIs from your code.</td>
</tr>
<tr>
<td>xGlobal</td>
<td>Method</td>
<td>terminalServer</td>
<td></td>
</tr>
<tr>
<td>xInfo</td>
<td>Method</td>
<td>directory</td>
<td></td>
</tr>
<tr>
<td>xInfo</td>
<td>Method</td>
<td>navPane</td>
<td></td>
</tr>
<tr>
<td>OBJECT</td>
<td>TYPE</td>
<td>NAME</td>
<td>NOTES</td>
</tr>
<tr>
<td>--------------</td>
<td>--------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>XmlDocument</td>
<td>Method</td>
<td>LoadSave</td>
<td></td>
</tr>
<tr>
<td>XmlWriter</td>
<td>Method</td>
<td>CreateNewFile</td>
<td></td>
</tr>
<tr>
<td>XppCompiler</td>
<td>Class</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This topic contains links to topics about developing user interface elements.

The user interface for Finance and Operation applications differs significantly from the interface for Microsoft Dynamics AX 2012. The client in Dynamics AX 2012 is a Microsoft Win32 application that has extensions that use ActiveX, WinForm, or WPF controls. The X++ application logic runs on the client for the form and table methods, and some logic occurs on the server. For controls, both the X++ logic application programming interface (API) and the physical Win32 control are tightly connected on the client. The client is an HTML web client that runs in all major browsers. These browsers include Microsoft Edge, Internet Explorer 11, Chrome, and Safari (see System requirements). The move to a web client has produced the following changes to client forms and controls:

- The physical presentation of forms and controls is now HTML, JavaScript, and CSS within the browser.
- Form controls are split into logical and physical parts. The X++ logical API and related state run on the server.
- The logical and physical parts are kept in sync through service calls that communicate changes from each side. For example, a user action on the client creates a service call to the server that is either sent immediately or queued so that it can be sent later.
- The server tier keeps the form state in memory while the form is open.

The form metamodel continues to be used to define controls and application logic. This approach supports almost all the existing Form, Form DataSource, and Form Control metamodel and X++ override methods. However, some control types, properties, and override methods have been removed, either because of incompatibility with the new platform or for performance reasons. For example, ActiveX and ManagedHost controls can no longer be used to add custom controls, because they are incompatible with the HTML platform. Instead, a new extensible control framework has been added that lets you add additional controls.

**Tutorials**

- Build the Rental Charge Type form
- Build the customer form

**Forms**

- Navigation concepts
- Page layout in the web client
- Dynamics Symbol font
- Test forms that use custom patterns

**Controls**

- Action controls
- Input controls and grid column sizes
- Check box support in tree controls
- Filtering options
- Display pages side-by-side using the Open in New Window icon
- Code migration - Context menu code
- Code migration - Mouse double-click logic
Messaging

- Slider and MessageBox
- Messaging API: Message center, Message bar, Message details
- Messaging the user

Form pattern guidelines

- Selecting a form pattern
- Form styles and patterns
- Form pattern add-ins

<table>
<thead>
<tr>
<th>FORM PATTERNS</th>
<th>SUPPORT FORM PATTERNS</th>
<th>SUB PATTERNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details Master</td>
<td>Advanced Selection</td>
<td>Custom Filter Group</td>
</tr>
<tr>
<td>Details Transaction</td>
<td>Dialog</td>
<td>Dimension Entry Control</td>
</tr>
<tr>
<td>Form Part Section List</td>
<td>Drop Dialog</td>
<td>Dimension Expression Builder</td>
</tr>
<tr>
<td>List Page</td>
<td>Lookup</td>
<td>Fields and Field Groups</td>
</tr>
<tr>
<td>Simple Details</td>
<td>Factbox</td>
<td>Filters and Toolbar</td>
</tr>
<tr>
<td>Simple List</td>
<td></td>
<td>Fill Text</td>
</tr>
<tr>
<td>Simple List and Details</td>
<td></td>
<td>Horizontal Fields and Buttons Group</td>
</tr>
<tr>
<td>Table Of Contents</td>
<td></td>
<td>Image Preview</td>
</tr>
<tr>
<td>Task Single</td>
<td></td>
<td>List Panel</td>
</tr>
<tr>
<td>Task Double</td>
<td></td>
<td>Nested Simple List and Details</td>
</tr>
<tr>
<td>Wizard</td>
<td></td>
<td>Section Chart</td>
</tr>
<tr>
<td>Workspace</td>
<td></td>
<td>Section Power BI</td>
</tr>
<tr>
<td>General Form Guidelines</td>
<td></td>
<td>Section Related Links</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section Stacked Chart</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section Tabbed List</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section Tiles Tabular Fields</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Toolbar and List</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Toolbar and Fields</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Workspace Filter Group</td>
</tr>
</tbody>
</table>

Control extensibility

- Building an extensible control
- Extensible control programming reference
- Control extensibility
- Create localizable labels
- Extensible control layout guidelines

- Contextual data entry for lookups
- HierarchyViewer control
- Lookup controls
- File upload control
- System-defined buttons
- Images on a page or in a grid
- Font and background colors for input, table, and grid controls
- Right-to-left language support and bidirectional text
- Create icons for workspace tiles
- Keyboard shortcuts for extensible controls
- Extensible controls – public JavaScript APIs
Control the text that Task Recorder generates for a control
Build the Rental Charge Type form
11/24/2021 • 5 minutes to read • Edit Online

In this lab you’ll create a Simple List form. A Simple List form can show reference or secondary data that has six or fewer fields. For example, the form that you create will list and describe the types of rental charges.

Prerequisites

For this tutorial, you’ll need to access the environment using Remote Desktop, and be provisioned as an administrator on the instance. For more information, see Access Instances.

Overview

To create the form, you’ll start from the existing form,FmtChargeType. This form uses the Simple List pattern. The following illustration shows the FmtChargeType form with the required controls from the Simple List pattern.

Adhering to the form pattern ensures that this Simple List form has the same structure and layout as other Simple List forms.

Key concepts

- Create a Simple List form using a pattern.
- Bind a table to the form.
- Add controls to the form.
- View the form using Visual Studio and a browser.

Setup

Import the tutorial project and transactional data

Use Visual Studio to import the tutorial project. The tutorial project includes the artifacts that you’ll use to complete this tutorial. Use Visual Studio to open the FMTutorial project and load the data for the tutorial. You’ll use the FMTDataHelper class to load data for the Fleet Management tutorial. If this is the first tutorial you’re working on, review Access Microsoft Instances and make sure you provision your administrator user if you’re working on a local VM.

1. Download the Fleet Management sample from https://github.com/Microsoft/FMLab, save it to C:\, and unzip it.

2. On the desktop, double-click the Visual Studio shortcut to open the development environment.
3. On the Finance and Operations menu, click Import Project.

4. In the Import Project window, next to the Filename text box, click the ellipsis button.

5. In the Select the file to import window, browse to C:\FMLab, click FMTutorialDataModel.axpp, and then click Open.

6. In the Project file location text box, enter C:\FMLab.

7. Select the Overwrite Elements option, and the Current solution radio button. The following illustration shows the completed Import Project dialog box.

8. Click OK.

9. In Solution Explorer, expand Classes, and under the FMTutorial project, right-click FMTDataHelper, and then click Set as Startup Object.

10. On the Build menu, click Rebuild Solution. Use the rebuild to make sure that all of the files in the project are built regardless of timestamps. You can view the build progress in the Output window.

11. After the build completes, press Ctrl+F5 to run the project. The browser will open and run the class that imports the data.

Open the FMTutorial project

Use Visual Studio to open the FMTutorial project. If you have Visual Studio open and have already loaded the FMTutorial project, you can continue to the next section.

1. If the development environment isn’t already open, on the desktop, double-click the Visual Studio shortcut to open the development environment.

2. On the File menu, click Open > Project/Solution.

3. In the Open Project dialog box, browse to C:\FmLab\FMTutorial, select the FMTutorial solution, and then click Open.

4. The FMTutorial project appears in Solution Explorer.

Use a template to create the form
Use Visual Studio to create the **FmtChargeType** form. You’ll use a template for building the Simple List form. You’ll also add a data source to the form and add fields to the data grid.

1. In **Solution Explorer**, right-click the **FMTutorial** project, point to **Add**, and then click **Existing Item**.

2. In the **Add Existing Item** window, browse to C:\FmLab, click **AxForm_FmtChargeType**, and then click **Add**. The **FmtChargeType** form appears at the bottom of the **FMTutorial** project in **Solution Explorer**.

3. In **Solution Explorer**, double-click **FmtChargeType**. The form opens in the Form designer.

4. Add the **FmtChargeType** table as the data source for the form. Right-click **Data Sources**, and then click **New Data Source**. A data source node is added.

5. Click the data source node from the previous step. In the **Properties** window, populate the following properties with the specified values.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>FMTChargeType</td>
</tr>
<tr>
<td>Name</td>
<td>FMTChargeType <em>Be sure to specify the value for the Table property first. This property will automatically update to use that same value.</em></td>
</tr>
</tbody>
</table>

The following illustration shows **Data Sources** after you add the **FMTChargeType** table.

6. In the Form designer, click **Design**. In the **Properties** window, populate the following properties with the specified values.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caption</td>
<td>Rental charge types <em>This is the label that appears at the top of the form.</em></td>
</tr>
<tr>
<td>Data Source</td>
<td>FMTChargeType <em>Use this property to specify the data source for the form.</em></td>
</tr>
</tbody>
</table>

7. In the Form designer, click **Design > Grid**.

8. Bind the **FMTChargeType** data source to the grid that appears in the simple list form. In the **Properties** window, in **Data Source**, enter **FMTChargeType**.
9. You have to specify the data source before you add fields to the grid. You can then use the fields from the data source to add columns to the grid. Add two fields from the data source to the grid. The fields you add will appear as columns on the Simple List form. Expand the FMTChargeType data source Fields node in the left pane. Press Ctrl and then click the following fields:
   - ChargeType
   - Description

10. Drag the selected fields to Design > Grid in the right pane. The following illustration shows the grid after the grid node is expanded and the two fields are added.

11. In the Form designer, click Design > CustomFilterGroup > QuickFilter.

12. In the Properties window, click TargetControl, and then select Grid to bind the QuickFilter control to the grid on the form.

13. Click File > Save FmtChargeType.

View the form

Use Visual Studio to build and run the FmtChargeType form.

1. In Solution Explorer, right-click the FmtChargeType form, and then click Set as Startup Object.

2. Press Ctrl+F5 to build and run the form.

3. The form opens in Internet Explorer.

4. To add a rental charge type, click New in the Action Pane at the top of the form. Add the following information.

<table>
<thead>
<tr>
<th>RENTAL CHARGE TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>cleaning</td>
<td>Cleaning fee</td>
</tr>
</tbody>
</table>

5. In the Action Pane, click Save.

6. Refresh the browser to see the new record in the list. The following illustration shows how the form should look.
7. The form opens in view mode. Click **Edit** in the Action Pane to switch the form into edit mode. To return to view mode, click **Options** and then **Read mode**.
In this lab you’ll create a Master Details form and apply the appropriate form pattern and subpatterns. A Master Details form shows primary data that has many fields. For example, the form that you create will show customer information.

Prerequisites

For this tutorial, you will need to access the environment using Remote Desktop, and be provisioned as an administrator on the instance. For more information, see Access Instances.

Overview

To create the form, you’ll start from the existing form, FmtCustomer. The form represents the old Master Details template. As a part of the tutorial, you’ll apply the Master Details pattern, which will enforce a consistent structure for this form type. The following illustration shows the FmtCustomer starting artifact.

Key concepts

- Create a Master Details form.
- Apply a form pattern to a form.
- Use the Visual Studio pattern add-ins to get information about form/model pattern coverage.
- Apply subpatterns to form controls.
- View the form using Visual Studio and a browser.
- Determine the amount of remaining patterns work in a model.

Setup

Import the tutorial project and transactional data

Use Visual Studio to import the tutorial project. The tutorial project includes the artifacts you will use to
Open the FMTutorial project

Use Visual Studio to open the FMTutorial project. If you have Visual Studio open and have already loaded the FMTutorial project, you can continue to the next section.

1. If the development environment is not already open, on the Desktop, double-click the Visual Studio shortcut.
to open the development environment.

2. On the File menu, click **Open** > **Project/Solution**.

3. In the **Open Project** dialog box, browse to C:\FmLab\FMTutorial, select the **FMTutorial** solution, and then click **Open**.

4. The FMTutorial project appears in **Solution Explorer**.

**Use a template to create the form**

Use Visual Studio to create the **FmtCustomer** form. You’ll use a template to create a new master details form. The data source for this tutorial is provided by the starter form. However, you’ll add fields to the grid and details view and apply the Master Details form pattern.

1. In **Solution Explorer**, right-click the **FMTutorial** project, point to **Add**, and then click **Existing Item**.

2. In the **Add Existing Item** window, browse to C:\FmLab, select **AxForm_FmtCustomer**, and then click **Add**. The **FmtCustomer** form appears at the bottom of the **FMTutorial** project in Solution Explorer.

3. In Solution Explorer, double-click **FmtCustomer**. The form opens in the form designer.

4. In the Form designer, click **Design**. In the **Properties** window, specify the following values.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source</td>
<td>FmtCustomer</td>
</tr>
<tr>
<td>Caption</td>
<td>Customers</td>
</tr>
</tbody>
</table>

5. In the Form designer, click **Design** > **GridDetailsTab** > **TabPageGrid** > **MainGrid**, and then click **MainGrid**.

6. In the **Properties** window, click **Data Source**, and then select **FmtCustomer** to bind the **FmtCustomer** table to the grid. You can now use the fields from the data source to add columns to the grid.

7. Click **Data sources** > **FmtCustomer** > **Fields** to add fields to the grid.

   a. Click **FirstName**, press and hold the Ctrl key, and then select the following additional fields in the order shown:

      - LastName
      - CellPhone
      - DriverLicense
      - Email

   b. Drag the highlighted fields to **Design** > **GridDetailsTab** > **TabPageGrid** > **MainGrid**. The following illustration shows the grid after expanding the grid node and adding the fields.
8. Click **Save**.

9. Click **Design > GridDetailsTab > TabPageDetails > TitleGroup** to add the record header to the details view.

10. Click **HeaderText**. In the **Properties** window, specify the following values.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source</td>
<td>FmtCustomer</td>
</tr>
<tr>
<td>Data Method</td>
<td>titleFields</td>
</tr>
</tbody>
</table>

11. Click **Design > GridDetailsTab > TabPageDetails > DetailsBodyTab > General** to add content to the details view.

a. Click **FmtCustomer > Data sources > FmtCustomer > Fields**, press and hold the Ctrl key, and then select the following fields:

   - FirstName
   - LastName
   - CellPhone
   - DriverLicense
   - Email

b. Drag the highlighted fields onto **General**, and then click **Save**.

**View the form**

Run the form to verify that it loads correctly.

1. In **Solution Explorer**, right-click **FmtCustomer**, and then click **Set as Startup Object**.

2. Press **Ctrl+F5**. The grid view should render like the following illustration.
Apply a pattern to the form

Use Visual Studio to apply the Master Details form pattern to the Customer form. Applying a form pattern ensures your form has the expected structure. It also simplifies the design experience by automatically setting the values of properties in the nodes that are part of the pattern.

1. Right-click Design, point to Apply pattern, and then click Details Master.

2. Add the missing Navigation List group. The red highlighting in the Patterns Information Panel indicates that this control is missing.
   a. Right-click Design, point to New, and then click Group.
   b. In the Properties window, in the Name property, enter SidePanel.
c. Click SidePanel, and press Alt+Up to move this group above the GridDetailsTab (Tab).

3. Click Design again. The yellow highlighting around the Navigation List and the Panel Tab indicate that there are problems that need to be resolved under each of these nodes before the pattern can be successfully applied.

4. In the Patterns Information Panel, click SidePanel.

5. Add the missing controls.
   a. Right-click SidePanel, point to New, and then click QuickFilter.
   b. In the Properties window, specify the following values.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>SidePanelQuickFilter</td>
</tr>
<tr>
<td>Target Control</td>
<td>MainGrid This QuickFilter should have the same columns available for filtering as the main grid on the form</td>
</tr>
</tbody>
</table>

c. Right-click SidePanel, point to New, and then click Grid.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>NavigationList</td>
</tr>
</tbody>
</table>
6. Add identifying fields to the Navigation list. Right-click **NavigationList**, point to **New**, and then click **String**.
   
a. In the **Properties** window, specify the following values.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataSource</td>
<td>FmtCustomer</td>
</tr>
<tr>
<td>DataMethod</td>
<td>fullName</td>
</tr>
<tr>
<td>Name</td>
<td>FmtCustomer_FullName</td>
</tr>
</tbody>
</table>

b. Expand **Data Sources** > **FmtCustomer** > **Fields** to add phone numbers to the Simple list.

c. Drag the **CellPhone** field onto the grid under **Design** > **SidePanel** > **NavigationList**.

7. Click **SidePanel**. Notice the **Patterns Information Panel** is now indicating that the controls in this subtree are in full compliance with the pattern.

![Patterns Information Panel](image)

8. Click **Design** > **GridDetailsTab**. The yellow highlighting around the subnodes indicates that there are problems that need to be resolved under both nodes before the form pattern can be successfully applied.

![Grid Details Tab](image)

9. Notice that the pattern expects the **Grid Panel** to be after the **Details Panel**. Click **TabPageGrid** and press **Alt+Down** to move that tab below the **Details Panel**.
10. Click GridDetailsTab. The TabPageDetails tab page now adheres to the pattern. However, the TabPageGrid tab page needs additional attention.

11. Click TabPageGrid. Focus in the designer is now on TabPageGrid, and the Patterns Information Panel has been updated.

12. The Patterns Information Panel now indicates a missing Group control at the top of the TabPageGrid container.
   a. Right-click TabPageGrid, point to New, and then click Group.
   b. Press Alt+Up two times to position the group as the first control in the group.
   c. In the Properties window, in the Name property, enter GridCustomFilterGroup.

13. The pattern is looking for a subpattern to be applied to GridCustomFilterGroup. Right-click GridCustomFilterGroup, point to Apply pattern, and then click Custom and Quick Filters.
14. The **Custom and Quick Filters** subpattern requires a QuickFilter control.
   a. Right-click **GridCustomFilterGroup**, point to **New**, and then click **QuickFilter**.
   b. In the **Properties** window, specify the following values.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>MainGridQuickFilter</td>
</tr>
<tr>
<td>Target Control</td>
<td>MainGrid</td>
</tr>
</tbody>
</table>

15. Browse up the control tree to design and notice how the **Patterns Information Panel** now shows no issues under each of the controls.

16. Press **Ctrl+S** to save the form.

**View the details form**

Run the form to see the Details view and the Grid view.

1. Press **Ctrl+F5** to run the project. The following illustration shows how the grid view appears.

2. Click **Phil** to go to the details view for that record.
3. Click the Show list button on the left side of the form to open the navigation list.

4. To go back to the grid view, click Close (or the browser Back button).

5. Return to Visual Studio.

### Add subpatterns

1. In Visual Studio, in the Form designer, right-click FmtCustomer, point to Addins, and then select Form statistics.

The Form Statistics add-in provides several useful data points about the state of the form. This includes:

- **Pattern=Unspecified count** – The number of nodes for which no form pattern or subpattern has been applied.

- **Pattern=Custom count** – The number of nodes for which a custom pattern was applied, meaning the structure did not fit with any existing pattern.

- **Pattern coverage** – The percentage of controls on the form that are covered by the form pattern or a subpattern. A value of 100% indicates a fully covered form.
2. To complete pattern coverage for this form, the Pattern=Unspecified count should be zero. Use the Visual Studio form search to find all instances of “unspecified” in the form.

3. Because the General tab page contains only input controls and no custom layout is required for this FastTab, the Fields and Field Groups pattern should be applied to guarantee a responsive layout. Right-click General, point to Apply pattern, and then select Fields and Field Groups.

4. On the far right of the screen, click Clear search.

5. Press Ctrl+S to save the form.

6. Repeat step 1 to run the Form Statistics add-in a second time to verify the form is fully covered by patterns.
7. Press Ctrl+F5 to run the project and see the updated form.

8. Click Adrian to go to the details view. The following illustration shows how the details view now appears after applying the Fields and Field Groups subpattern so that the fields lay out responsively. By changing the browser width, you’ll see how the field layout adjusts to better fill the width of the browser.

9. Return to Visual Studio

Determine the amount of remaining patterns work in a model

1. Click Finance and Operations, point to Addins, and then select Run form patterns report.

A notification dialog will be shown when the form patterns report has been generated.
2. Open the PatternsReport file in Excel.

3. Filter the report to the Fleet Management Tutorial model.
   a. Click Data > Filter.
   b. Filter the Model column to FleetMgmtTutorial.

The report shows pattern-related information regarding the forms in this model including the top-level form pattern currently applied, and the percentage of controls on the form covered by patterns. This can be used to track the remaining patterns work in one or more models.
In this tutorial, you will add navigational elements to a workspace and the navigation pane.

**Prerequisites**

For this tutorial, you need to access the environment using Remote Desktop, and be provisioned as an administrator on the instance. For more information, see [Access Instances](https://docs.microsoft.com).

**Key concepts**

- A **workspace** is an overview page that is specific to a particular subject area. Workspaces are common to all users. In this tutorial, you will add content into an existing workspace.
- The **dashboard** is the default home page for each user.
- **Tiles** are securable objects that can be shown on a workspace or the dashboard. They can be secured by using menu items.

**Setup**

If this is the first tutorial that you are working on, review [Access Instances](https://docs.microsoft.com) and make sure that you provision your administrator user if you are working on a local VM.

**Import the tutorial project**

If you have already imported the Fleet management tutorial project, skip to the next section.

1. Download the Fleet Management sample from [https://github.com/Microsoft/FMLab](https://github.com/Microsoft/FMLab), save it to C:\, and unzip it.
2. In Visual Studio, on the **Finance and Operations** menu, click **Import Project**.
3. In the **Import Project** window, next to the **Filename** text box, click the ellipsis button.
4. In the **Select the file to import** window, browse to C:\FMLab, click **FMTutorialDataModel.axpp**, and then click **Open**.
5. In the Project file location text box, enter C:\FMLab.
6. Select the **Overwrite Elements** option, and then click **OK**.

**Import transactional data**

1. In Visual Studio, open the **FMTutorial** project. On the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, browse to C:\FMLab\FMTutorial, and then click **FMTutorial**. Click **Open**. The **FMTutorial** project appears in **Solution Explorer**.
3. Use the FMTDataHelper class to load data for the Fleet Management tutorial. In **Solution Explorer**, in the **FMTutorial** project, expand **Classes**, right-click **FMTDataHelper**, and then click **Set as Startup Object**.
4. From the **BUILD** menu, click **Rebuild Solution**. You use the rebuild to update the timestamps of the imported artifacts. You can view the build progress in the **Output** window.
5. Press Ctrl+5 to run the project and load the data.

**Add a tile to the tutorial workspace**

First, we will add a new tile to the form FMTClerkWorkspace.
1. In Solution Explorer, expand Forms and then double-click FMTClerkWorkspace.

2. In the designer, expand PanoramaBody.

3. Right-click TileContainer, and then click New > Tile Button.

4. Specify the following properties for the new tile button.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Test tile</td>
</tr>
<tr>
<td>Tile</td>
<td>FMTAllCustomersTile</td>
</tr>
</tbody>
</table>

   This will create a duplicate of the existing All customers tile.

5. In Solution Explorer, click Forms > FMTClerkWorkspace, right-click, and then select Set as Start-up Object. Setting a start-up object is necessary to allow Visual Studio to launch when you press Ctrl+F5 in step 7. Setting this form as the start-up object will cause the work-in-progress Fleet management clerk workspace to appear after you press Ctrl+F5. We will preview this form again later in detail.

6. Right-click FMTutorial, and then click Rebuild.

7. Press Ctrl+F5 to run the project.

   After you build and run the project, the Fleet management clerk workspace will launch. The new tile named, Test tile, that you created will be included in the first section of the workspace, at the end of the set of tiles.

   The tile will not navigate anywhere when clicked. To enable this, you can define a Menu Item Name on FMTAllCustomersTile, under Tiles in Solution Explorer.
Add a new workspace to the navigation pane

Next, we will add the FMTClerkWorkspace form to the navigation pane. We will do this in two locations:

- The All workspaces list.
- A new item in the area list containing a menu structure that shows the workspace.

Create a menu item that points to the FMTClerkWorkspace workspace

1. Right-click FMTutorial, point to Add, and then click New Item.
2. Click AX Artifacts > User Interface > Display Menu Item. In the Name property, enter FMTClerkWorkspace.
3. Click Add.
4. Specify the following properties for the new menu item.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Reservation management tutorial</td>
</tr>
<tr>
<td>Object</td>
<td>FMTClerkWorkspace</td>
</tr>
</tbody>
</table>

Create a tile that points to the FMTClerkWorkspace workspace menu item

1. Right-click FMTutorial, point to Add, and then click New Item.
2. Click AX Artifacts > User Interface > Tile. In the Name property, enter FMTClerkWorkspace.
3. Click Add.
4. Specify the following properties for the new tile.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MenuItemName</td>
<td>FMTClerkWorkspace</td>
</tr>
</tbody>
</table>

Add a menu extension for the navigation pane

1. In Application Explorer, click User Interface > Menus, right-click NavPaneMenu, and then click Create extension.
2. In Solution Explorer, double-click NavPaneMenu.Extension.
3. In the designer, right-click NavPaneMenu.Extension, point to New, and then click Submenu.
4. Select the new submenu. In the Name property, enter NavPaneMenuFleetTutorial.
5. In Solution Explorer or Application Explorer, locate the FMTClerkWorkspace tile, and drag it onto the newly created submenu. Click Save.
6. Right-click FMTutorial, and then click Rebuild.
7. Press Ctrl+F5 to run the project. After you build and run the project, the navigation pane will contain a link to the new workspace. Open the navigation pane by clicking the navigation pane button (three lines) at the top right of the application window.
8. When you open the navigation pane, select **All workspaces**, and scroll down in the list after it opens. You should see the following new Reservation management tutorial workspace in the list.

Add the form to the main menu structure

Now you’ll add a new main menu section that contains a tile that points to the tutorial workspace. You will then add a link to the same form in this section. This will demonstrate the appearance of a non-workspace form link.

1. In Visual Studio, in **Solution Explorer**, right-click **FMTutorial**, point to **Add**, and then click **New Item**.

2. Click **AX Artifacts** > **User Interface** > **Menu**. In the **Name** property, enter `FleetManagementTutorial`.

3. Click **Add**.

4. In **Solution Explorer**, double-click the new menu **FleetManagementTutorial** if it isn’t already open.

5. In the properties list, set the **Label** property to `Fleet management tutorial`.

6. In the designer, right-click **FleetManagementTutorial**, and click **New > Submenu**.

7. Specify the following properties for the new submenu.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Workspaces</td>
</tr>
<tr>
<td>Label</td>
<td>Workspaces</td>
</tr>
</tbody>
</table>

8. In **Solution Explorer** or **Application Explorer**, locate the **FMTClerkWorkspace** display menu item and drag it onto the new **Workspaces** submenu.

9. In the designer, right-click **FleetManagementTutorial**, and then click **New > Submenu**.

10. Specify the following properties for the new submenu.
11. In Solution Explorer or Application Explorer, locate the FMTClerkWorkspace display menu item and drag it onto the new Common submenu.

12. In Application Explorer, click User Interface > Menus > MainMenu. Right-click MainMenu, and then click Create extension.

13. In Solution Explorer, locate and open the new extension. Select and double-click MainMenu.Extension to open it.

14. In the designer, right click MainMenu.Extension, point to New, and then click Menu reference.

15. Specify the following properties for the new menu reference.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>FleetManagementTutorial</td>
</tr>
<tr>
<td>Menu Name</td>
<td>FleetManagementTutorial</td>
</tr>
</tbody>
</table>

16. Click Save.

17. Right-click FMTutorial, and then click Build.

18. Press Ctrl+F5 to run the project.

19. Go to the main menu section you just modified. Open the navigation pane and scroll down until you see the new top-level Fleet management tutorial menu. You may need to clear your browser cache by pressing Ctrl+F5.
20. Click **Fleet management tutorial** > **Workspaces** to expand that submenu. Your navigation pane should look like the following.

If you click on the **Common** submenu, you will see the menu item that you modeled there. You can click either of these links to check that you have set up the references correctly. If you have set up the references correctly, the tutorial workspace you’re working on should open when clicked on.
In this tutorial, you will create a new tile and include it in the summary section of a workspace, build a new list for a workspace, and create a data cache for the list in the workspace.

**Prerequisites**

For this tutorial, you must access the environment by using Remote Desktop, and you must be provisioned as an administrator on the instance. For more information, see [Deploy and access development environments](#).

**Key concepts**

- Learn about and use form patterns that are related to workspaces.
- Create a new tile, and include it in the **Summary** section of a workspace.
- Build a new list for a workspace.
- Create a data cache for the list in the workspace.

**Setup**

**Import the tutorial project and transactional data**

Use Microsoft Visual Studio to import the tutorial project. The tutorial project includes the artifacts that you will use to complete this tutorial. Use Visual Studio to open the FMTutorial project and load the data for the tutorial. You will use the FMTDataHelper class to load data for the Fleet Management tutorial. If this is the first tutorial that you’re working on, review [Deploy and access development environments](#), and make sure that you provision your administrator user if you’re working on a local virtual machine (VM).

1. Download the FMTutorialDataModel.axpp file from the Microsoft Dynamics Lifecycle Services (LCS) methodology, and copy it to the Downloads folder of the VM.
2. On the desktop, double-click the Visual Studio shortcut to open the development environment.
3. On the Dynamics 365 menu, click Import Project.
4. In the Import Project dialog box, next to the File name field, click the ellipsis (...) button.
5. In the Select the file to import dialog box, browse to the Downloads folder, click FMTutorialDataModel.axpp, and then click Open.
6. Select the Overwrite Elements check box and the Current solution option. The following illustration shows the completed Import Project dialog box.
Exercise 1: Understand the operational workspace pattern

Before you start to make adjustments to FmtClerkWorkspace form, you will look at the current state of the form to better understand what content is already there and how that content fits the Operational Workspace pattern.

1. In Solution Explorer, double-click the FmtClerkWorkspace form to open it in the designer.

2. Click the Design node.

3. Click the Pattern tab. Operational workspaces have an optional Action Pane and optional filter group (as indicated by the 0..1 notation to the left of those nodes). However, the panorama-style tab is required by this pattern. The Patterns tab shows that the PanoramaBody control matches the required Tab in the pattern, but there are no corresponding controls for the optional items at this level of the pattern.
All workspaces have three required sections:

- **Summary section** – This section is intended to contain tiles or form parts, which correspond to a card or chart.
- **Tabbed list** – This section consists of one or more lists of data that is relevant to the user’s work. Only one list is shown at a time, and each list can optionally include local filters and actions. Individual lists are modeled inside form part controls.
- **Related links** – This section consists of important or commonly used links for this activity or persona.

Operational workspaces can optionally include a panorama section that contains up to two charts (the Section Charts tab page) and a Power BI section.

**View the workspace**

1. In Solution Explorer, right-click the FmtClerkWorkspace form, and then click **Set as Startup Object**.

2. Press **Ctrl+F5** to build and run the form. The form opens in Internet Explorer.
Exercise 2: Create a new tile for the workspace

Now that you understand the content structure of a workspace, you will see how to add content to a workspace. For example, one important piece of information for this workspace might be the number of rentals that are currently in progress. In this section, you will add the required metadata to add a new tile to the Summary section of the FmtClerkWorkspace form to show this information. To make this tile work correctly, you will have to add four metadata artifacts: a query, a menu item, a tile, and a tile button.

Add a query that retrieves current rentals

All tiles require a backing query to retrieve the correct information.

1. In Solution Explorer, in the FMTutorial project, right-click the Queries folder, point to Add, and then click New item.

2. Click Dynamics 365 Items > Data Model > Query. For the Name property, enter FMTRental_Current.

3. Click Add.

4. If the new FMTRental_Current query isn’t already open in the designer, double-click it in Solution Explorer.

5. In the designer, right-click Data Sources, and then click New Data Source.

6. In the Properties window, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>FMTRental</td>
</tr>
<tr>
<td>Dynamics Fields</td>
<td>Yes</td>
</tr>
</tbody>
</table>

7. Right-click Ranges, and then click New Range.

8. In the Properties window, set the following properties.
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>State</td>
</tr>
<tr>
<td>Value</td>
<td>InProgress</td>
</tr>
</tbody>
</table>

9. Right-click **Order By**, and then click **New Field**.

10. In the **Properties** window, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direction</td>
<td>Descending</td>
</tr>
<tr>
<td>Data Source</td>
<td>FMTRental</td>
</tr>
<tr>
<td>Field</td>
<td>StartDate</td>
</tr>
</tbody>
</table>

11. Press **Ctrl+S** to save.

**Add the corresponding menu item**

1. In Solution Explorer, in the **FMTutorial** project, right-click the **Menu items** folder, point to **Add**, and then click **New item**.

2. Click **Dynamics 365 Items** > **User Interface** > **Display menu item**. Set the **Name** property to **FMTRental_Current**.

3. Click **Add**.

4. If the new **FMTRental_Current** menu item isn’t already open in the designer, double-click it in Solution Explorer.

5. In the **Properties** window, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>@FMT197 This value corresponds to “Current rentals”.</td>
</tr>
<tr>
<td>Object</td>
<td>FMTRental</td>
</tr>
<tr>
<td>Query</td>
<td>FMTRental_Current</td>
</tr>
</tbody>
</table>

6. Press **Ctrl+S** to save.

**Add a tile**

1. In Solution Explorer, in the **FMTutorial** project, right-click the **Tiles** folder, point to **Add**, and then click **New item**.

2. Click **Dynamics 365 Items** > **User Interface** > **Tile**. Set the **Name** property to **FMTRental_Current**.

3. Click **Add**.

4. If the new **FMTRental_Current** tile isn’t already open in the designer, double-click it in Solution Explorer.

5. In the **Properties** window, set the following properties.
Add a tile button to the workspace form

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>ShortWide</td>
</tr>
<tr>
<td>Menu Item Name</td>
<td>FMTRental_Current</td>
</tr>
<tr>
<td>Type</td>
<td>Count</td>
</tr>
</tbody>
</table>

Press Ctrl+S to save.

Tiles also have a refresh frequency property that controls how often the counts on the tiles are automatically updated. The value that is set for this property should be based on the demand for updated counts, together with query execution speed against volume data. For guidance about how to set this property, see Tile and List Caching for Workspaces. For this lab, the default value of 10 minutes will be enough.

**Add a tile button to the workspace form**

1. In Solution Explorer, double-click the FmtClerkWorkspace form to open it in the designer.

2. Right-click Design > PanoramaBody > TileContainer, point to New, and then click Tile Button.

3. Press Alt+Up arrow four times to move the tile button to the top of TileContainer.

4. In the Properties window, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>FMTCurrentRentalsTile</td>
</tr>
<tr>
<td>Tile</td>
<td>FMTCurrentRentalsTile</td>
</tr>
</tbody>
</table>

5. Press Ctrl+S to save.

**View the new tile on the workspace**

Use Visual Studio to build and run the updated FmtClerkWorkspace form.

1. In Solution Explorer, right-click the FmtClerkWorkspace form, and then click Set as Startup Object.

2. Press Ctrl+F5 to build and run the form. The form opens in Internet Explorer.
3. Click the **Current rentals** tile. You go to the **Rentals** page, which should be filtered to the three current rentals.

4. Click the **Back** button or the **Close** button to return to the workspace.

5. Click on the small `i` button in the upper-right corner of the **Current rentals** tile. You see information about how current the data in the tile is. Additionally, a link is provided that you can use to manually refresh the tile to view updated data.

**View tile data cache values at run time**

A system administrator can modify tile cache parameters at run time by using the **Tile data cache configuration** page.

1. Click in the navigation search field on the navigation bar.

2. Type **Tile data**, and then click **Tile data cache configuration** in the search results.

3. Find the **FMTCurrentRentalsTile** record.

   From this page, the system administrator can perform several run-time modifications to a tile cache. For example, the system administrator can enable/disable the data cache, modify the refresh frequency, and
enable/disable the ability to manually refresh the count tile. Note that tile caches are registered when a form that has a tile is first opened. Therefore, the list of tiles that is shown in your environment might differ from the list in the preceding illustration.

Exercise 3: Create a new tabbed list in the workspace

Next, you will next see how to include an additional list in the workspace. This section will give you some experience with building forms and will also expose you to form patterns. You will add a list of available vehicles, so that you will be able to initiate a new rental by selecting an available vehicle. In this section, you will just add the new list to the new workspace but won’t add the action to initiate the rental. To add this list, you will have to complete the following tasks:

1. Add a new tab page to the workspace.
2. Add a new form that has the list content.
3. Add a new menu item that points to the new form.
4. Add a new query to limit the vehicles to available vehicles.

Add space in the workspace for a new list

1. In Solution Explorer, double-click the FmtClerkWorkspace form to open it in the designer.
2. Right-click Design > PanoramaBody > TabbedListSection > TabbedLists, and then click New Tab Page.
3. In the Properties window, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>AvailableVehiclesContainer</td>
</tr>
<tr>
<td>Caption</td>
<td>@FMT199 This value corresponds to “Available vehicles”.</td>
</tr>
</tbody>
</table>

4. Right-click AvailableVehiclesContainer, point to New, and then click Form Part. Form Part is the only control type that the Operational Workspace pattern allows here. This control will be used to link to the form that you will build to hold the content for this section.

5. In the Properties window, set the Name property to AvailableVehiclesPart.
6. Press Ctrl+S to save.

Add a new form that has the new workspace content

1. In Solution Explorer, in the FMTutorial project, right-click the Forms folder, point to Add, and then click New item.
2. Click Dynamics 365 Items > User Interface > Form. Set the Name property to FMTAvailableVehicles.
3. Click Add.
4. If the new **FMTAvailableVehicles** form isn't already open in the designer, double-click it in Solution Explorer.

5. Add the **FmtVehicle** table as a data source for the form.
   a. Right-click **Data Sources**, and then click **New Data Source**.
   b. Click the new data source node. In the **Properties** window, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>FMTVehicle</td>
</tr>
<tr>
<td>Name</td>
<td>FMTVehicle <strong>Note:</strong> Be sure to specify the value for the <strong>Table</strong> property first. This property is automatically updated so that it uses the same value.</td>
</tr>
</tbody>
</table>

6. Add the **FmtVehicleModel** table as a second data source for the form.
   a. Right-click **Data Sources**, and then click **New Data Source**.
   b. Click the new data source node. In the **Properties** window, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>FMTVehicleModel</td>
</tr>
<tr>
<td>Name</td>
<td>FMTVehicleModel <strong>Note:</strong> Be sure to specify the value for the <strong>Table</strong> property first. This property is automatically updated so that it uses the same value.</td>
</tr>
<tr>
<td>Join Source</td>
<td>FMTVehicle</td>
</tr>
<tr>
<td>Link Type</td>
<td>Inner Join</td>
</tr>
</tbody>
</table>

7. Notice the **Pattern: <select>** notation next to **Form Design**. This indicates the required pattern for this node. Right-click **Design**, point to **Apply pattern**, and then click **Form Part Section List**. This form pattern is typically used by workspace lists.

8. Click the **Pattern** tab to see the expected content for this pattern. This information will help guide you as you create content for the form. **Note:** In the future, we plan to provide a mechanism for automatically creating a form structure, based on a selected form pattern.

   ![Pattern Diagram](image)

   In particular, this pattern looks for the following elements:

   - An optional header group that contains any filters and actions that are required for this workspace list.
- A required grid. As the red border in the preceding illustration indicates, the patterns engine can't currently find this element.
- An optional default action, which can provide navigation to a backing form for an individual record in the grid.
- An optional button that takes the user to a backing form that shows the full list of items in this section.

9. Right-click **Design**, point to **New**, and then click **Grid**.

10. In the **Properties** window, set the following properties. **Note:** The grid that we are building will contain cards and will be two cards wide.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExtendedStyle</td>
<td>cardList</td>
</tr>
<tr>
<td>Multi Select</td>
<td>No</td>
</tr>
<tr>
<td>Style</td>
<td>List</td>
</tr>
<tr>
<td>Data Source</td>
<td>FMTVehicle</td>
</tr>
<tr>
<td>Name</td>
<td>VehicleList</td>
</tr>
<tr>
<td>Visible Columns Mode</td>
<td>Fixed</td>
</tr>
<tr>
<td>Visible Columns</td>
<td>2</td>
</tr>
</tbody>
</table>

11. Right-click the **VehicleList** grid, point to **New**, and then click **Group**.

12. In the **Properties** window, set the **Name** property to **VehicleCard**.

13. Right-click the **VehicleCard** group, point to **Apply pattern**, and then click **Business Card – Three Fields**.

14. Right-click the **VehicleCard** group, point to **New**, and then click **Image**.

15. In the **Properties** window, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source</td>
<td>FMTVehicleModel</td>
</tr>
<tr>
<td>Data Method</td>
<td>vehicleImage</td>
</tr>
</tbody>
</table>

16. Expand **Data Sources > FMTVehicle > Fields**.

17. Drag the **VehicleId** and **DisplayRelationType** fields into the **VehicleCard** group.

18. Right-click **Design**, point to **New**, and then click **Group**. Update the new group's **Name** property to **HeaderGroup**.

19. The new HeaderGroup element requires a subpattern, because there are two variants for the arrangement of filters and actions in these sections. Right-click **HeaderGroup**, point to **Apply pattern**, and then click **Filters and Toolbar – Inline**.

20. Right-click **HeaderGroup**, point to **New**, and then click **Group**. Update the new group's **Name** property to **FilterGroup**.
21. Right-click **FilterGroup**, point to **New**, and then click **QuickFilter**.

22. In the **Properties** window, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>VehicleQuickFilter</td>
</tr>
<tr>
<td>Target Control</td>
<td>VehicleList</td>
</tr>
</tbody>
</table>

23. Press **Ctrl+S** to save.

**Add a new query that limits the data to available vehicles**

1. In Solution Explorer, in the **FMTutorial** project, right-click the **Queries** folder, point to **Add**, and then click **New item**.

2. Click **Dynamics 365 Items > Data Model > Query**. Set the **Name** property to **FMTAvailableVehicles**.

3. Click **Add**.

4. If the new **FMTAvailableVehicles** query isn’t already open in the designer, double-click it in Solution Explorer.

5. In the designer, right-click **Data Sources**, and then click **New Data Source**.

6. In the **Properties** window, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>FMTVehicle</td>
</tr>
<tr>
<td>Dynamics Fields</td>
<td>Yes</td>
</tr>
</tbody>
</table>

7. Right-click **Ranges**, and then click **New Range**.

8. In the **Properties** window, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Status</td>
</tr>
<tr>
<td>Value</td>
<td>Available</td>
</tr>
</tbody>
</table>

9. Press **Ctrl+S** to save.

**Add a new menu item that references the new form**

1. In Solution Explorer, in the **FMTutorial** project, right-click the **Menu items** folder, point to **Add**, and then click **New item**.

2. Click **Dynamics 365 Items > User Interface > Display menu item**. Set the **Name** property to **FMTAvailableVehicles**.

3. Click **Add**.

4. If the new **FMTAvailableVehicles** menu item isn’t already open in the designer, double-click it in Solution Explorer.
5. In the **Properties** window, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>FMTAvailableVehicles</td>
</tr>
<tr>
<td>Query</td>
<td>FMTAvailableVehicles</td>
</tr>
</tbody>
</table>

6. Press **Ctrl+S** to save.

**Link the new list to the workspace**

1. In Solution Explorer, double-click the **FmtClerkWorkspace** form to open it in the designer.
2. Click **Design > PanoramaBody > TabbedListSection > TabbedLists > AvailableVehiclesContainer > AvailableVehiclesPart**.
3. In the **Properties** window, set the **Menu item name** property to **FmtAvailableVehicles**.

**View the new menu item**

Use Visual Studio to build and run the updated **FmtClerkWorkspace** form.

1. In Solution Explorer, right-click the **FmtClerkWorkspace** form, and then click **Set as Startup Object**.
2. Press **Ctrl+F5** to build and run the form. The form opens in Internet Explorer.
3. Click the **Available vehicles** tab to see the new list.
4. Click in the QuickFilter, type **Lit** and then press **Enter** to filter down to Litware model vehicles that are available.

**Exercise 4: Create a backing data cache for a list**

For lists that have expensive queries, or lists where multiple users might use and work from the same workspace, consider caching the list to improve performance. In this section, you will add the necessary artifacts to create a data cache for a list. These artifacts include a query that maps fields to the cache, a table that holds the cache, and a class that provides the mapping between the query and the table. You will then uptake this cache on one of the tabbed lists in the workspace.

**Add a query that maps fields to the data cache**

The first step is to build a query that will be used to populate the cache table. This query should include all the tables that you want to get your cache data from, and it should limit the results to those records/columns that you want cached.

1. In Solution Explorer, in the **FMTutorial** project, right-click the **Queries** folder, point to **Add**, and then click **New item**.
2. Click Dynamics 365 Items > Data Model > Query. Set the Name property to FMTPickupAndReturnQuery.

3. Click Add.

4. If the new FMTPickupAndReturnQuery query isn’t already open in the designer, double-click it in Solution Explorer.

5. In the designer, right-click Data Sources, and then click New Data Source.

6. In the Properties window, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>FMTRental</td>
</tr>
<tr>
<td>Dynamics Fields</td>
<td>No</td>
</tr>
</tbody>
</table>

7. Add the following five fields to the FMTRental data source. For each field, right-click Fields, point to New, and then click Field. In the Properties window, set the Field property as appropriate.

- StartDate
- EndDate
- Vehicle
- State
- RentalId

8. Right-click Ranges, and then click New Range.

9. In the Properties window, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>State</td>
</tr>
<tr>
<td>Value</td>
<td>1..2 Note: This value will cache rentals that are Ready for Pickup or In Progress.</td>
</tr>
</tbody>
</table>

10. Under FMTRental, right-click Data Sources, and then click New Data Source.

11. In the Properties window, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>FMTCustomer</td>
</tr>
<tr>
<td>Dynamics Fields</td>
<td>No</td>
</tr>
</tbody>
</table>

12. Add the following three fields to the FMTCustomer data source. For each field, right-click Fields, point to New, and then click Field. In the Properties window, set the Field property as appropriate.

- FirstName
- LastName
- Image

13. Right-click Relations, and then click New Relation.

14. In the Properties window, set the following properties.
The query that you’ve constructed should match the following illustration.

15. Press Ctrl+S to save.

**Add a cache table**

The second step is to define a table that has the fields that are returned from the cache query. You must also add a `SysDataCacheContextId` field that will be used to map the cache row to the base framework cache tables. Additionally, you should also define any required relations between this table and other tables, and also any data methods that you require that involve the cached fields.

1. In Solution Explorer, in the FMTutorial project, right-click the Tables folder, point to Add, and then click New item.

2. Click Dynamics 365 Items > Data Model > Table. Set the Name property to FMTPickupAndReturnTableCache.

3. Click Add.

4. If the new FMTPickupAndReturnTableCache table isn’t already open in the designer, double-click it in Solution Explorer.

5. In the designer, right-click Fields, and then add the following fields. For each field, the following table shows the data type and the extended data type (EDT) or enum type.
<table>
<thead>
<tr>
<th>FIELD TYPE</th>
<th>FIELD NAME</th>
<th>EDT/ENUM TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>FirstName</td>
<td>FirstName (EDT)</td>
</tr>
<tr>
<td>String</td>
<td>LastName</td>
<td>LastName (EDT)</td>
</tr>
<tr>
<td>Container</td>
<td>Image</td>
<td>Bitmap (EDT)</td>
</tr>
<tr>
<td>Int64</td>
<td>Vehicle</td>
<td>FMTVehicleRecId (EDT)</td>
</tr>
<tr>
<td>Utc Date Time</td>
<td>StartDate</td>
<td>StartDateTime (EDT)</td>
</tr>
<tr>
<td>Utc Date Time</td>
<td>EndDate</td>
<td>EndDateTime (EDT)</td>
</tr>
<tr>
<td>Int64</td>
<td>SysDataCacheContextId</td>
<td>SysDataCacheContextId (EDT)</td>
</tr>
<tr>
<td>Enum</td>
<td>State</td>
<td>FMTReservationState (Enum)</td>
</tr>
<tr>
<td>String</td>
<td>RentalId</td>
<td>FMTRentalId (EDT)</td>
</tr>
</tbody>
</table>

6. In the designer, right-click Relations, point to New, and then click Relation.

7. In the Properties window, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>FMTRental</td>
</tr>
<tr>
<td>Related Table</td>
<td>FMTRental</td>
</tr>
</tbody>
</table>

8. Right-click the FMTRental relation, point to New, and then click Normal.

9. In the Properties window, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>RentalId</td>
</tr>
<tr>
<td>Related Field</td>
<td>RentalId</td>
</tr>
</tbody>
</table>

10. In the designer, right-click Mappings, and then click New.

11. In the Properties window, set the Map property to SysDataSetCacheTableMap.

12. Click Id. In the Properties window, set the Map Field To property to RecId.

13. Click SysDataCacheContextId. In the Properties window, set the Map Field To property to SysDataCacheContextId. **Note:** If the field doesn't appear in the list, you might have to save the table first, by pressing Ctrl+S.

14. Press F7 to view the table's code. Alternatively, right-click FMTReturnAndPickupTableCache, and then click View Code.

15. Add the following display methods to the table. The form will use these methods later.
Adding a cache class that links the query and table

The third step is to create a class that defines the relationship between the cache query and the catch table.

1. In Solution Explorer, in the FMTutorial project, right-click the Classes folder, point to Add, and then click New item.

2. Click Dynamics 365 Items > Code > Class. Set the Name property to FMTPickupAndReturnClass.

3. Click Add.

4. If the new FMTPickupAndReturnClass class isn't already open in the designer, double-click it in Solution Explorer.

5. Add the following code to the class.

```csharp
public display FMTName fullName()
{
    return this.FirstName + ' ' + this.LastName;
}
public display container customerImage()
{
    ImageReference imgRef;
    container imgContainer = this.Image;
    if(imgContainer == connull())
    {
        imgRef = ImageReference::constructForSymbol("Person");
        imgContainer = imgRef.pack();
    }
    return imgContainer;
}
public display str rentalVehicle()
{
    FMTVehicle vehicle;
    str value;
    if(this.Vehicle == 0)
    {
        value = "No vehicle assigned";
    }
    else
    {
        select vehicle where vehicle.RecId == this.Vehicle;
        value = vehicle.Description;
    }
    return value;
}

16. Press Ctrl+S to save.

```
After you’ve set up the data cache, you can start to use the cache in your forms. In this section, you will update one of the workspace lists so that it uses the data cache.

1. In Solution Explorer, double-click the FMTReturningTodayPart form to open it in the designer.
2. Expand the Data Sources node.
3. Delete the FMTCustomer data source.
4. Click the FMTRental data source. In the Properties window, set the Table property to FMTPickupAndReturnTableCache.
5. Click Design > ReturningTodayGrid. In the Properties window, set the Data Source property to FMTPickupAndReturnTableCache.
6. Inside ReturningTodayGrid, click CustomerImage. Update the Data Source property to FMTPickupAndReturnTableCache and the Data Method property to customerImage.
7. Inside ReturningTodayGrid, click FirstNameCopy1. Update the Data Source property to FMTPickupAndReturnTableCache and the Data Method property to fullName.
8. Press F7 to view code for the form.
9. Instrument the form so that it can react to data caching, as shown in the following code.

```java
public class FMTReturningTodayPart extends FormRun implements SysIDataSetConsumerForm {
    public void registerDatasourceOnQueryingEvent() {
        FMTPickupAndReturnTableCache_DS.OnQueryExecuting +=
            eventhandler(this.parmDataSetFormQueryEventHandler().prepareDataSet);
    }
}
```

10. Press Ctrl+S to save.
Update the action above the list so that it works with the cache table

Actions that are performed from the workspace might expect records from the base tables. Therefore, these actions might have to be updated so that they work with the cache table. In this example, the FmtCompleteRecord form currently expects a FMTRental record as context. Therefore, this form must be updated so that it works correctly with either a base rental record or a cache table record as context.

1. In Solution Explorer, double-click the FmtCompleteRental form to open it in the designer.

2. Press F7 to view the form's code.

   ```java
   public void init()
   {
       // If this form was opened with a Rental as context
       if (element.args() != null && element.args().record() != null && element.args().record().TableId == tablenum(FMTRental))
       {
           // Get the Rental context
           rentalDS = FormDataUtil::getFormDataSource(element.args().record());
           rental = element.args().record();
           if (rental != null)
           {
               select firstonly forupdate vehicle where vehicle.RecId == rental.Vehicle;
           }
       }
       super();
   }
   ```

3. Update the init() method so that it matches the following code.

   ```java
   public void init()
   {
       // If this form was opened with a record context
       if (element.args() != null && element.args().record() != null)
       {
           // Get that context
           rentalDS = FormDataUtil::getFormDataSource(element.args().record());
           if (element.args().record().TableId == tableNum(FMTPickupAndReturnTableCache))
           {
               FMTPickupAndReturnTableCache cacheRecord = element.args().record();
               select firstonly forupdate rental where rental.RentalId == cacheRecord.RentalId;
           }
           else if (element.args().record().TableId == tableNum(FMTRental))
           {
               rental = element.args().record();
           }
           if (rental != null)
           {
               select firstonly forupdate vehicle where vehicle.RecId == rental.Vehicle;
           }
       }
       super();
   }
   ```

4. Press Ctrl+S to save.

Update the Returning today query so that it works with the cache table

1. In Solution Explorer, double-click the FmrRental_ReturningToday query to open it in the designer.

2. Expand the Data Sources node, and then click FMTRental.

3. In the Properties window, update the Table property to FMTPickupAndReturnTableCache. Note: The Name property should be updated automatically to the same value.

4. Press Ctrl+S to save.
View the updated query
Use Visual Studio to build and run the updated FmtClerkWorkspace form.

1. In Solution Explorer, right-click the FmtClerkWorkspace form, and then click Set as Startup Object.
2. Press Ctrl+F5 to build and run the form. The form opens in Internet Explorer.
3. Click the Returning today vertical tab.
4. Click Complete rental for the second record in the list.
5. Set End Mileage to 200, and then click OK. Notice that the rental that you just returned still appears in the list.

Make sure that your workspace is responsive
You must make sure that your lists remain up to date after a user performs an action that should remove a record from the list (for example, the user completes a rental). In this section, we will instrument that action to help guarantee that the workspace reacts appropriately.

1. In Solution Explorer, double-click the FmtCompleteRental form to open it in the designer.
2. Press F7 to view the code for the form.
3. Locate the clicked() code for OKButton. Near the end of this method is a research call on the calling form’s data source. Just before that line of code, add the following if statement to delete the processed rental from the cache table.

   ```csharp
   if(rentalDS.table() == tableNum(FMTPickupAndReturnTableCache))
   {
       //Delete updated record from backing cache
       FMTPickupAndReturnTableCache cacheRecord = element.args().record();
       cacheRecord.delete();
   }
   rentalDS.research(true);
   }
   ```
4. Press Ctrl+S to save.

View the updated form
Use Visual Studio to build and run the updated FmtClerkWorkspace form.

1. In Solution Explorer, right-click the FmtClerkWorkspace form, and then click Set as Startup Object.
2. Press Ctrl+F5 to build and run the form. The form opens in Internet Explorer.
3. Click the Returning today vertical tab.
4. Click Complete rental for the second record in the list.
5. Set End Mileage to 100, and then click OK. Notice that the rental that you just returned no longer appears in the list.

Related tutorials
- **Build the Customer form** – See this tutorial if you want more exposure to form patterns. This tutorial will walk through the process of applying the Details Master pattern to a form.
- **Build navigation** – See this tutorial if you want instructions for adding your workspace to the menu structure.
This article describes the primary navigation concepts including the dashboard, the new navigation search feature, the navigation pane, workspaces, and tiles.

Navigation concepts

The primary navigation concepts are:

- Dashboard
- Navigation pane
- Workspaces
- Tiles
- Navigation search

The dashboard is a new concept, whereas the navigation pane and workspaces are updates to existing concepts. To implement the navigation concepts, the user interface model uses several standard page types. When you create an application, you should follow the conventions for these pages to present a consistent experience for the user. The following diagram shows an overview of the standard page types and how they fit together.

The following sections provide more detail about the pages that underlie these concepts. They include information about the modeling of these and other types of pages.

Dashboard

The dashboard is the first page that users see when they access the client. The dashboard contains tiles that show important details from the system. Content that was previously displayed in Cues on Role Center pages in Microsoft Dynamics AX 2012 is now available on the dashboard. You can return to the dashboard at any time by clicking **Dynamics 365** on the navigation bar at the top of the application frame.

The dashboard primarily consists of a large section of workspace tiles. There might also be a Getting Started tool, which isn’t shown in the preceding screenshot. The dashboard’s workspace tile section is built from a menu structure that has its root in the **NavPaneMenu** menu. The menu is modified by a set of menu extensions, and those extensions contain one or more tile references that correspond to the tiles that users see in that section.
Navigation pane

The navigation pane provides access to workspaces, main menu elements, recently opened forms, and user-defined favorites. The user can open the navigation pane by clicking the **Show navigation pane** button under the navigation bar. The navigation pane consists of four collapsible sections. The **Favorites** section provides quick access to the list of forms the user has explicitly marked as a favorite. Marking a form as a favorite is accomplished by clicking the star icon next to the form in the navigation pane. The **Recent** section lists the forms the user has most recently visited. The set of workspaces a user has access to is conveniently shown in the **Workspaces** section. Finally, the **Modules** section provides the full list of modules. Clicking on a module will open the right side of the navigation pane, where the user can navigate to the desired page in that module.

**Note:** In this screenshot, **All customers** has been marked as a favorite, and therefore it will appear in the **Favorites** list.

Like the workspace tiles on the dashboard, the elements that are listed in the navigation pane are generated at runtime, based on a menu structure. The same root menu (**NavPaneMenu**) that defines the set of workspaces on the dashboard also defines the navigation pane. Here's an example of the logical structure for the navigation pane:

- **NavPaneMenu** (menu)
  - **NavPaneMenuFleet** (menu extension)
    - "Reservation management" (tile reference)

Workspaces

Workspaces are activity-oriented pages that are designed to increase a user's productivity by providing information that answers the targeted user's most pressing activity-related questions and allows the user to initiate their more frequent tasks. Access to the various workspaces depends on the roles that users have in the organization. Much of the list and business intelligence (BI) content from the old Role Center pages is exposed on workspaces. To navigate to a workspace, you can click a tile on the dashboard, click a link in the navigation pane, or find the workspace using the navigation search feature (see the next section).

Workspaces are simply a different type of modeled forms. The system differentiates workspaces by **Form.Design.Style=Workspace** (this value is defined as part of the Workspace form pattern). A workspace consists of a caption (which is defined on **Form.Design.Caption**) and a panorama (that is, a tab control). A panorama contains sections of content that are relevant to the task for which the workspace is intended. These sections are modeled as tab pages. The first section will generally be a set of tiles that users can click to begin
new tasks or access lists of items. The second section contains a set of relevant lists for the activity. The last section contains a number of links to pages that are important but not frequently used for this activity. In between the list and links section are a few optional sections that might contain charts and graphs. One important distinction of workspaces is that they do not have a data source. If the content (such as a list or chart) requires a data source, you must model that content on an independent form of type Form part, and then reference that form part on the workspace. Form parts can be hosted on other forms, and each can have its own data source.

Tiles

Windows 8 introduced the concept of tiles, and you will see them used in the client. A tile is a rectangular button that behaves like a menu item button. It is used to navigate to or open pages. In addition, tiles can display relevant data, such as counts or key performance indicators (KPIs). A tile can include images that provide the user with additional visual context. You can create the following types of tiles.

<table>
<thead>
<tr>
<th>TYPE</th>
<th>EXAMPLE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td><img src="image" alt="Customers report" /></td>
<td>This type of tile does not show any business data.</td>
</tr>
<tr>
<td>Count</td>
<td>30 All rentals</td>
<td>This type of tile shows a count of the items in the referenced form's query. Note that the tile in this example uses the ShortWide size.</td>
</tr>
<tr>
<td>KPI</td>
<td><img src="image" alt="Total revenue" /></td>
<td>This type of tile shows a summary of data from a KPI.</td>
</tr>
<tr>
<td>Link</td>
<td><img src="image" alt="Rentals report" /></td>
<td>This type of tile points to a URL. The tile has the same appearance as a tile of the Standard type.</td>
</tr>
</tbody>
</table>

For modeling, you first create a tile element, which is an abstract element. The tile element is then referenced on a form by a tile button element. Multiple tile button elements can refer to a single tile. Tiles are defined first by the Type property (which has valid values of Standard, Count, KPI, and Link). After you specify the type, you specify the following minimum properties for each type:

- **Standard** and Count:
  - Label
  - Menu Item Name
- Menu Item Type
- **KPI:**
  - Label
  - KPI
- **Link:**
  - Label
  - URL

After these properties are defined, your tile is complete. To make the user experience richer, you can optionally extend tiles of the Standard and Link types so that they include images. Use the following properties to add images.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Location</td>
<td>See the information about basic button behavior in <strong>Actions.</strong></td>
</tr>
<tr>
<td>Normal Image</td>
<td>See the information about basic button behavior in <strong>Actions.</strong></td>
</tr>
<tr>
<td>Tile Display</td>
<td>Define how the image appears on the tile.</td>
</tr>
</tbody>
</table>

The following are the valid options and corresponding behaviors for the **Tile Display** property.

<table>
<thead>
<tr>
<th>VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto</td>
<td>The behavior is the same as the behavior for <strong>TextAndImage.</strong></td>
</tr>
<tr>
<td>TextAndImage</td>
<td>The tile shows the specified label and the image in a small container above it. For example, the <strong>App links</strong> tile uses this value.</td>
</tr>
<tr>
<td>TextOnly</td>
<td>The other two image-related properties are ignored, and only the label is shown.</td>
</tr>
<tr>
<td>ImageOnly</td>
<td>The label is ignored, and only the image is shown.</td>
</tr>
<tr>
<td>BackgroundImage</td>
<td>The specified image is expanded to fill the tile from edge to edge, and the label is overlaid on the image in the same location. Shading is applied behind the text to ensure that it remains legible.</td>
</tr>
</tbody>
</table>

**Note:** The following limitations apply when the **BackgroundImage** value is used:

- If a symbol is used, it isn’t displayed.
- The image that you specify isn’t resized. Therefore, you must create an image of the appropriate size to guarantee that it fills the tile correctly. Currently, a standard-sized tile is a square that is 130 pixels on each side.

**Navigation search**

There is a convenient search mechanism for finding and navigating to forms and workspaces that appear in the navigation pane and on the dashboard. For example, a search on the keywords “all sales order” returns a list of navigation elements that match those keywords.
The search keywords are matched not only to the caption of the navigation elements but also to the corresponding path. For example, a search on the keywords "ven bal report" returns results that match "vendor balance" in the caption and "report" in the path.
This topic discusses layout in the web client. Layout is a design process that specifies how controls appear on a page.

**Introduction**

Layout is a design process that specifies how the controls on a page appear in the web client. Layout occurs within container controls. The following table lists the container controls.

<table>
<thead>
<tr>
<th>CONTAINER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form.Design</td>
<td>The root of the page. It functions as a special kind of container.</td>
</tr>
<tr>
<td>Group</td>
<td>The general-purpose container control. Group controls can be nested as required.</td>
</tr>
<tr>
<td>Tab</td>
<td>A control that contains TabPage controls and has many possible Tab.Style values, such as Tab, FastTab, Vertical Tab, and Panorama.</td>
</tr>
<tr>
<td>TabPage</td>
<td>The appearance of each TabPage control depends on its Tab.Style value.</td>
</tr>
<tr>
<td>ButtonGroup</td>
<td>A special type of Group control that contains buttons.</td>
</tr>
</tbody>
</table>

A grid is a special type of control that has some container behaviors, such as flexible sizing (SizeToAvailable). However, a grid has special visualizations and isn’t a general-purpose container control.

**Layout: Dynamics AX 2012 vs. Finance and Operations apps**

**Layout in Dynamics AX 2012**

In Microsoft Dynamics AX 2012, the arrangement of controls in containers is almost always vertical, and columns are manually set to provide some horizontal spread.

**Examples**

Columns=1 1 2 3 Columns=2 1 4 2 5 3 In Dynamics AX 2012, sizing is achieved via the Height and Width properties. If Height and Width are set to Auto, the size is as large as the child controls require. If Height and Width are set to Column, the container is as large as it can be within the parent container. By default, Height and Width are set to Auto for every container.

**Layout in Finance and Operations**

In Finance and Operations, layout is controlled by the same basic properties that control layout in Dynamics AX 2012. However, additional options have been added to support a more responsive layout. In particular, the layout of a page is based on the following factors:

- The arrangement method that is specified by the ArrangeMethod property.
- The columns that are specified by the Columns property.
- The sizing that is specified by the HeightMode, WidthMode, Height, and Width properties.
ArrangeMethod property

The **ArrangeMethod** property specifies a base arrangement method for a container. For this property, Finance and Operations apps have all the options from AX 2012. However, they also have a **HorizontalWrap** option that is intended for tile layouts in panoramas. The following table describes the various options for the **ArrangeMethod** property.

<table>
<thead>
<tr>
<th>OPTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertical</td>
<td>Controls are arranged vertically. If columns are also used, controls are arranged vertically inside the generated columns. This option is the default value for Groups and for TabPages where Tab.Style is set to a value other than Panorama.</td>
</tr>
<tr>
<td>HorizontalLeft</td>
<td>Controls are arranged horizontally, and they are left-aligned and bottom-aligned inside the parent container.</td>
</tr>
<tr>
<td>HorizontalRight</td>
<td>Controls are arranged horizontally, and they are right-aligned and bottom-aligned inside the parent container.</td>
</tr>
<tr>
<td>HorizontalWrap</td>
<td>Controls are arranged inside columns of fixed width that wrap horizontally. This option is typically used for tile layouts in panorama sections. It's the default value for TabPages where Tab.Style is set to Panorama.</td>
</tr>
</tbody>
</table>

ColumnsMode property

For the **ColumnsMode** property, Finance and Operations apps have a **Fill** option to support responsive layouts. When the property is set to this value, columns automatically flow as required. The following table describes the various options for the **ColumnsMode** property.

<table>
<thead>
<tr>
<th>OPTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fill</td>
<td>Columns are generated to fill the available horizontal space or vertical space, depending on the container type. If the container is a Panorama-style tab, this option generates columns to fill it along the vertical axis. For all other containers (Groups, Tab-style Tabs, and all other styles of Tabs), this option generates columns to fill the container along the horizontal axis.</td>
</tr>
<tr>
<td>Fixed</td>
<td>Specify the number of columns that the Columns property should generate. Controls are evenly distributed among the columns, and their order is maintained. If the controls can't be distributed evenly among the columns, the leftmost columns receive extra controls first. This option is the default value for all controls.</td>
</tr>
</tbody>
</table>

HeightMode/WidthMode properties

In Finance and Operations apps, sizing is done via two pairs of size properties: **WidthMode** and **Width**, and **HeightMode** and **Height**. The following table describes the various options for these properties.
<table>
<thead>
<tr>
<th>OPTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SizeToAvailable</td>
<td>Fill the available space along the vertical (or horizontal) axis inside the parent container. If the parent container has SizeToContent height (or width), the child's height (or width) is also SizeToContent, unless there is a sibling in the container that can provide a height (or width). This option is the default value for Grids and Tabs (of all styles).</td>
</tr>
<tr>
<td>SizeToContent</td>
<td>The height (or width) of the container should be the height (or width) of its contents. This option is the default value for Groups and all other controls except Tabs. FastTabs that aren't always expanded also have SizeToContent height.</td>
</tr>
</tbody>
</table>
| Manual               | The height (or width) is manually sized. Set HeightMode (or WidthMode) to Manual, and then set Height (or Width) to a fixed number of pixels.  
Note: Microsoft doesn't recommend that you use manual heights and widths, because they don't adapt to changes in form density. |

Note that if a value of Auto is used for these properties, the behavior is automatically determined at runtime. Typically, a value of Auto for these properties causes the same behavior as a value of SizeToContent, as in AX 2012.

Interactions between the ArrangeMethod and Columns properties

If ArrangeMethod=HorizontalLeft or HorizontalRight, the Columns property has no effect, because items are laid out in strict horizontal arrangement and no wrapping is used. If ArrangeMethod=Vertical, columns are arranged vertically, and the controls are either distributed evenly among the columns (Fixed), or distributed to fill the available horizontal or vertical space (Fill). If ArrangeMethod=HorizontalWrap, columns are arranged, and horizontal wrapping is used at a fixed column width of 280 px. Typically, this option is used to wrap tile layouts.

**Examples**

**ArrangeMethod=HorizontalWrap**

**ArrangeMethod=HorizontalWrap and Columns=1**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

**ArrangeMethod=HorizontalWrap and Columns=2**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>
For this example, we assume that only three lines of items can fit in the container height.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>7</th>
<th>8</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>9</td>
<td>10</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>11</td>
<td>12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ArrangeMethod=Vertical and Columns=Fill

For this example, we assume that only three lines of items can fit in the container height.

<table>
<thead>
<tr>
<th>1</th>
<th>4</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

ArrangeMethod=Vertical and Columns=Fill on a FastTab

For this example, we assume that the width of the FastTab can fit four columns.

<table>
<thead>
<tr>
<th>1</th>
<th>3</th>
<th>5</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
</tr>
</tbody>
</table>

Breakable groups
When you set `ColumnsMode` to Fill to dynamically create columns, based on the amount of available space, groups of fields can be split into multiple columns. The Breakable property on Group controls lets developers ensure that controls in a group aren't distributed across columns. The default value for this property is Yes, which indicates that the contents of the group can be split between groups. To keep a group together all the time, set Breakable to No. Note that Breakable applies only to the first level in nested groups.

**Guidelines for using layout properties**

**ColumnsMode=Fill**

- Don't nest containers that have `ColumnsMode` set to Fill. Set `ColumnsMode` to Fill only on the direct parent container of the controls/fields that you want to responsively fill the available space.
- Don't set HeightMode to SizeToAvailable on any child controls of a container that has `ColumnsMode` set to Fill. `ColumnsMode=Fill` tries to calculate an average height of all controls across columns to balance them as much as possible. However, our layout CSS and calculations can't handle SizeToAvailable children, and this setting doesn't necessarily make sense.
- Don't set WidthMode to SizeToAvailable on any child controls of a container that has `ColumnsMode` set to Fill. Otherwise, the child controls will take up all the available width, and all controls will appear in one column.
- Container that have `ColumnsMode=Fill` should have WidthMode set to SizeToAvailable (if you're using fill-width containers such as Tabs and Groups) or HeightMode set to SizeToAvailable (if you're using fill-height containers such as Panorama sections).

**HeightMode/WidthMode=SizeToAvailable**

- If you use WidthMode=SizeToAvailable, make sure that parent containers in the form have WidthMode set to SizeToAvailable, not SizeToContent. SizeToAvailable containers inside SizeToContent containers are overridden and become SizeToContent containers.

**Additional resources**

[User interface development home page]
The Dynamics Symbol font defines the set of out-of-box symbols that are available in the product. These symbols are primarily used for buttons, tiles, and image controls. In every release, there might be updates to this font. For example, symbols might be added or removed.

To access the list of available symbols (the name and an image) for every release that updated the Symbol font, visit the Dynamics Symbol Font page. A description of the various locations where symbols are used in the product and usage guidelines for each location is also included.
Introduction

Personalization plays an important role in allowing users and organizations to optimize the user experience to meet their needs. For more details on personalization, see Personalize the user experience.

Traditional personalization let users have only one set of personalizations per page. The Saved views feature expands on personalization in several important ways:

- Views permit users to have multiple named sets of personalizations per form, which they can quickly switch between as needed. This allows a user to create multiple optimized views of a page, where each view has been tailored to fit the needs of performing a particular business task.

- Views created for particular page types can also include user-added filters or sorts, which allows users to quickly return to commonly filtered datasets. See the What pages support views section for more details.

- Views can be published to users in specific security roles and specific legal entities. Therefore, any user who has a specified role and access to a specified legal entity can access and use that view, even if that user doesn’t not have permission to personalize. This publish capability lets organizations define corporate, standard views that are optimized for their business. For more information, see the Managing personalizations at an organizational level with views section.

- Unlike traditional personalization, views aren’t automatically saved when a user performs personalizations or filters a list. Explicit saves are required to give users the flexibility to create a view before or after the changes that are associated with that view have been made. This requirement also ensures that view definitions aren’t unintentionally changed by filters or personalizations that aren’t intended for long-term use. Items that the system automatically stores as part of typical page usage (for example, column widths, or the expanded or collapsed state of sections) will be saved per view.

- Views can be added to workspaces as tiles, lists, or links. Therefore, a filtered data set can be surfaced in a workspace, and users can associate a set of personalizations that is relevant to that data set with a tile or link.

Switching between views

After views have been made available for an environment, the top of any page that supports views will include a collapsed view selector control that shows the name of the current view.

There are two size variations to the view selector:

- Large view selectors – Pages that prominently feature a list will have a larger view selector for a few reasons. Most importantly, the larger view selector indicates the pages where the view can include user-defined filters. Because filters are included in the views, the larger selector size is also warranted as the view names will often be the best description of the data shown on the screen and the expectation is that users will switch between views more often on these page types.

- Small view selectors – All other full-screen pages (except workspaces and the dashboard) have a smaller view selector that appears next to the page caption. Views on these pages include only personalizations, not user-defined filters. On these pages, the caption or record title is often the most important information at the top of the page. The smaller size of the view selector also reflects the lower frequency of view switching that is expected on these pages.

If you select the view name, the view selector is opened and shows the list of available views for the page.

Version 10.0.21 or later: If the Improved legal entity support for saved views feature is turned on, the
Creating and modifying views

Unlike traditional personalization, views aren't automatically saved when a user personalizes the page, or when a user applies a filter to a list or sorts it. An explicit action is required to save these changes to a view. This requirement gives users the flexibility to create a view before or after the changes that are associated with that view have been made. It also ensures that view definitions aren't unintentionally changed by one-time filters or personalizations. Note that typical page usage items (for example, column widths, or the expanded or collapsed state of sections) are automatically saved to the current view, even for locked views.

To ensure that the current state of the view is known, when you start to change a view by personalizing or filtering it, an asterisk (*) appears next to the current view name. This symbol indicates that you're looking at an unsaved, modified version of that view.

If you want to save those changes, follow these steps.

1. Select the view name to open the view selector.
2. To modify the existing view, select Save. Note that this action isn't available for locked views.
3. To create a new view:
   a. Select Save as.
   b. In the Save view as pane, enter a name and, optionally, a description for the view.
   c. If you want this view to be your default view, select Pin as default. For more information about default views see the Changing the default view section that follows.
   d. Version 10.0.21 or later: If the Improved legal entity support for saved views feature is turned on, you can select whether you want this view to be available for all legal entities or just a subset of them.
   e. Select Save.

Changing the default view

The default view is the view that the system tries to open when you first open the page. You should set the default view to the view that you expect to use most often.
**NOTE**

- In the base *Saved views* feature, there is a single, global default view across legal entities. If you change the default view, that view will be opened by default, regardless of the legal entity that you're currently in.
- **Version 10.0.21 or later:** When the *Improved legal entity support for saved views* feature is turned on, each legal entity can have its own default view per page.

To change the default view for a page, follow these steps:

1. Switch to the view that you use as the default.
2. Select the view name to open the view selector.
3. Select *More* and then *Pin as default*.

Alternatively, when you create a new view (by using the *Save as* action), you can make that new view the default view by setting the *Pin as default* option before you save the view.

**WARNING**

In some cases, the query that is associated with the default view isn't run when you first open a page. For example, if you open the page through a tile, the tile's query will be run, regardless of the query that is associated with the default view. Additionally, if you open a page that has a *standard* view that already has a defined query, the original query will be run instead of the default view's query. In this case, you will receive an informational message when the view is loaded. If you switch views after the page has been loaded, the view query should be able to be run as expected. In version 10.0.10 and later, the informational message that you receive will have an embedded action that lets you load the default view's query directly.

---

**Managing personal views**

The *Manage my views* dialog box gives you basic maintenance capabilities over your personal views and the order of views in the view selector. To open this page, select the view name to open the view selector drop-down menu, select *More*, and then select *Manage my views*.

**Version 10.0.21 or later:** If the *Improved legal entity support for saved views* feature is turned on, the *My views* section of the *Manage my views* dialog box shows the available views for the page in sections. Any views that are specific to the current legal entity are shown in their own section. The *Global views* section is always shown, so that you can manage the views that are available for the page in all legal entities.

For a list of available views for that page, the following set of actions are available.

- **Change the default view** – Use the *Pin as default* action to make the currently selected view the default view for this page. If the *Import legal entity support for saved views* feature is turned on, the *Global views* section lets you make a view the default view for either the current legal entity or all legal entities.
- **Reorder your views** – Use the *Move up* and *Move down* actions to rearrange your views in a specific order.
- **Rename a view** – Use the *Rename* action to change the name of the currently selected personal view. This action is turned off for locked views.
- **Delete a view** – Use the *Delete* action to permanently delete the currently selected view from the page. There is no way to recover a view after you remove it.

Any changes made in this dialog box will take effect after you select the *Update* button.

---

**Managing personalizations at an organizational level with views**

To help you understand how saved views help improve management of personalizations at an organizational
level, this section describes some differences in personalization management with and without the Saved views feature.

Without views, administrators would apply a set of personalizations for a page to a user or a group of users via the Personalization page. If those users had personalization rights, the personalizations would be applied to that page. However, there was no ability to prevent users from further personalizing the page, which meant the organization could not ensure that its users had a consistent user interface. If any of those users didn’t have personalization rights, the personalizations given to them by an administrator were not loaded. Further, if new users were hired into an organization, administrators needed to manually load a set of personalizations for the user. There was no automatic mechanism for specifying that a certain set of personalizations should be available for users in that role.

The Saved views feature makes organizational management of personalizations much easier, primarily because views can be published to groups of users. After a view has been published, any user who has one of the defined security roles and access to one the specified legal entities can see and use the view, even if that user doesn’t have access to personalization. Although every user has a copy of the published view, where page usage items are automatically applied, no user can save personalizations or query updates to a published view. In other words, published views are locked. Additionally, if new users are assigned to roles in legal entities that views were published to, they will automatically see the views that are associated with their roles and legal entities. No additional action is required by the admin. Likewise, if users change roles in an organization or are given access to different legal entities, they might no longer be able to access the views that were previously published to them. Again, no additional action is required by the admin.

Updates to a published view can easily be distributed to users by republishing the view to the appropriate security roles and legal entities.

The publish capability allows organizations to define corporate standard views that are optimized for their business, targeted at users in specific security roles.

Publishing views

During the publishing process, views can be assigned to one or more security roles for one or more legal entities. Therefore, any user who has access to a legal entity and is assigned to one of those roles can access and use the views. However, the user can’t edit the views. By default, system admins have access to the Publish action in the view selector drop-down menu. However, other trusted users in your organization can also be given access to view publishing via the new Saved views administrator role.

To publish a view, follow these steps:

1. Create and save a personal copy of the view that you want to publish.

2. With that view currently loaded, select the view name to open the view selector drop-down menu.

3. Select the More button and then select Publish. The Publish dialog box will open.

4. Enter a name for the view. The name that you enter is the name that users who receive this view will see in their view selectors. The names of published views for a page must be unique. No duplicate names are allowed, even if the list of roles or legal entities that the views are applied to differ.

5. Update 10.0.17 or later: If the (Preview) Translation support for organization views feature is turned on, you can add translations for your view name in as many languages as your organization requires by selecting the Translations button next to the Name field. The view name will then be shown to users in their current language. You can also set the default language to specify the translation that will be shown to users who are running languages that no translation is defined for.

6. Optional: Enter a description for the view, so that users who receive this view can better understand its purpose.
7. Determine whether the view should be published as the default view for the selected users. When a view is made the default view, users will see it the next time that they open the target page. The single, global default view of every targeted user will be changed. However, users can still change their default view after publishing has occurred.

**NOTE**

Be aware of the following behavior when you publish a view as the default view:

- If you publish a view as the default view to some or all legal entities, the follow behavior occurs:
  - If only the base Saved views feature is turned on, the single, global default view will be changed for every targeted user.
  - Version 10.0.21 or later: If the Improved legal entity support for saved views feature is turned on, and you publish the view to a subset of legal entities, the default view for those legal entities will be changed for every targeted user.
- If a user has roles where multiple views are published as the default view, the last view that was published will be used as the user's default view.

8. Add the security roles that correspond to the users who are being targeted by this view.

9. Determine whether you want to publish the view to the child roles of each security role that is selected. If you do, select the Include child roles check box in the row for the appropriate security roles. Note that this check box isn't available for roles that don't have child roles.

10. Add the legal entities that this view should be available for.

**NOTE**

Be aware of the following behavior if you publish a view to a specific legal entity, but you don't publish that view as the default view:

- If only the base Saved views feature is turned on, the user's view selector for the page initially shows the view only for the specified legal entities. However, after the view is loaded for the first time, the view selector for the page will always show it, regardless of the legal entity.
- Version 10.0.21 or later: If the Improved legal entity support for saved views feature is turned on, the view selector will only ever show the view for the specified legal entities.

11. Select Publish.

Note that in some environments, it may take some time (up to an hour) before users see the published view.

**Modifying a published view**

After you publish a view, you might find that you want to change it. Although you can't make live changes to a published view, because these views are locked for editing for all users (including publishers), you can republish a view to update it.

If the changes that you want to make to a published view only involve the publish parameters (the name and description of the view, or the security roles the view is published to), do the following:

1. Switch to the published view for the parameters that you want to update.
2. On the view selector drop-down menu, select Republish. If you're using version 10.0.12 or earlier, you must select Publish and then Yes to update the existing view.
3. Update the name, description, security roles, and legal entities for the view.
4. Select Publish. If you originally selected this published view as the default view, it will be the default view for
Managing published views

Like managing personal views, the Manage my views dialog box gives users with publish privileges basic maintenance capabilities over that page’s published views (in addition to their own personal views). To open this page, select the view name to open the view selector drop-down menu, select More, and then select Manage my views.

Although all users have a My views tab that show their personal views, users who have publish privileges also have an Organization views tab that shows all the published and unpublished views for that page. Because several users might be publishing views, it’s important that you be able to manage the full list of published views, even if you aren’t the user who published a given view.

For the list of all published views for the page, the following set of actions are available.

- Republish – Use the Republish action to republish a view after publishing parameters (name, description, security roles, or legal entities) are changed.
- Publish – Use the Publish action to publish a view that is currently unpublished.
- Unpublish – Use the Unpublish action to make a view inactive. The view will still be available in the system, but users won’t see it in the view selector until the view is published again.
- Save as personal – Use the Save as personal action to create a personal draft copy of the published view. This capability can help you understand the contents of a view that wasn’t published to you or that hasn’t yet been published. You can also use it to edit and then republish a view.
- Delete – Use the Delete action to permanently delete a published or unpublished view. This action also removes the view for all users in the system. The removal of published views takes effect after the Save button is selected. After a view is deleted, it can’t be recovered.

Managing views globally

Although some management capabilities are surfaced on every page, as indicated in this topic, system administrators and saved view administrators can manage views more holistically for the system via the Personalization page. In particular, this page has the following sections and capabilities:

- Published views – This section lists all views that have been published for your organization. From here, you can republish a view after you adjust the security roles or legal entities that the view targets. You can also export, delete, or unpublish views. You can use the Save as personal action to create a personal copy of a view, so that you can update the view or gain a better understanding of its contents.
- Unpublished views – This section lists all the organization views in your system that aren’t currently published. These views most often come into the system through the import capability. You can publish, export, or delete these views. The Quick publish action that was added in version 10.0.12 enables multiple views from this section to be published in one action, by using the existing security role and legal entity configurations. You can use the Save as personal action to create personal copies of these views, so that you can gain a better understand their contents.
- Personal views – This section lists all views that have been created by users in the system. From here, you
can publish a personal view to the organization, or copy one or more of these views to other users. You can also export or delete these views as required.

- **User settings** – Select a user to view, or adjust the user’s ability to use personalization either for the whole system or for specific pages that the user has visited. You can view and interact with the user’s personalizations in the system. You can also delete all personalizations for that user or reset feature callouts for the user. If feature callouts are reset, any pop-up windows that introduced new features and that the user previously dismissed will appear again the next time that the user encounters those features.

- **System settings** – You can temporarily turn off personalization for all users in the system. In this case, no personalizations are applied for any user, and all pages are reset to their default state. If you turn personalization back on later, all personalizations are reapplied. You can also permanently delete all personalizations for all users in the system. Personalizations that have been deleted can’t be recovered.

Therefore, before you perform this task, be sure to export any personalizations that you might want later.

Users who have access to the **Personalization** page can also import personal or organization views by using the **Import views** button on the Action Pane. For organization views, you can select **Publish immediately** to make the views available to users without an additional explicit publish.

### Known issues

For a list of known issues with saved views, please see [Build forms that fully utilize saved views.](#)

### Frequently asked questions

#### How do I enable saved views in my environment?

**NOTE**

The **Saved views** feature requires the Personalization system in Finance and Operations to be enabled. If personalization is turned off for the entire environment, views will be disabled even if you follow steps below.

You can turn the **Saved views** feature on and off through Feature management in any environment. After it’s turned on, saved views will be enabled in all subsequent user sessions.

#### What happens to existing personalizations when views are enabled?

When views are enabled, any existing personalizations for a user and form are saved into a new view called **My view** that is automatically set as the default view. This is meant to ensure that there is a consistent user experience before and after views are enabled, except for the view selector control appearing on forms.

#### What pages support views?

Views are available on most, but not all pages. Specifically, views are currently available on all full-screen pages except for dashboards and workspaces. Non-full-screen pages, which include dialog boxes, drop-down dialogs, lookups, enhanced previews, currently do not support views. View support for additional page types, such as workspaces and dialog boxes, may be considered for a future update.

#### Who is allowed to publish views?

Only system admins and users who have been assigned to the **Saved views administrator** role have the rights to publish views.

#### Why am I not able to save filters with this view?

There are a few reasons why a filter may not appear to save with a view:

- The page may not support saving filters as part of the view definition. Note that only pages with large view selectors allow personalizations and query modifications to be saved as a view. See the [Switching views](#) section for more information.
The page in question may not properly support views, as it may ignore the view query completely or may operate on a temporary table whose data is not persistent.

**What data will I see when I visit a page?**

For pages that have small view selectors (only personalizations can be saved to the view), you will see the same data as you always have when you visit the page.

For pages that have large view selectors (both personalizations and queries can be saved to the view), you will typically see the data that is linked to the query that is associated with your default view. There are two main exceptions:

- If you navigate to a page from a tile, the tile query will execute regardless of the query associated with the default view. If you created that tile after views have been enabled, selecting a tile will open the page with the view associated with that tile.
- If you navigate to a page and that entry point includes a query, the original query will execute originally in place of the default view's query. You should be alerted when this occurs via an informational message when the view is loading. You can also confirm by switching to this view after the page loads, as that should allow the view query to execute regardless.

**Why is a view that was published for a specific legal entity visible in all legal entities?**

If you publish a view to a specific legal entity, but you don't publish that view as the default view, the following behavior occurs:

- If only the base Saved views feature is turned on, the user's view selector for the page initially shows the view only for the specified legal entities. However, after the view is loaded for the first time, the view selector for the page will always show it, regardless of the legal entity. This behavior occurs because users get their own personal copy of the published view when it's loaded, and personal views are global.
- **Version 10.0.21 or later:** If the Improved legal entity support for saved views feature is turned on, the view selector will only ever show the view for the specified legal entities. This behavior occurs because the feature enables views (including personal views) to be linked to specific legal entities.
Saved views are an important expansion of personalization capabilities Finance and Operations applications. While the Saved views topic provides general details about this feature, this topic focuses on the more technical elements of saved views as well as aspects of form development that may be impacted by views.

"User-perceived" pages

Traditionally, a set of personalizations has a 1:1 link to a modeled form. For many pages, this makes sense to the user, as the user’s perception of the page matches the way in which the form is modeled. However, in some cases the 1:1 link of a modeled form to a set of personalizations is not intuitive or obvious because users do not see or care about the boundaries between modeled forms.

Saved views try to eliminate this confusion by letting users create views on “user-perceived” pages. Therefore, users don’t have to understand how forms are modeled to understand how and when personalizations are applied. Consider the following two scenarios:

- **More than one “user-perceived” page in a single modeled form**: The standard modeling of Master Details and Transaction Details forms (for example, the CustTable and PurchTable forms, respectively) consists of more than one “user-perceived” page: a grid page and a details page.

  Because users are not aware when this transition from list to details crosses a form boundary (nor do they need to know this), view support in Details forms is handled differently to allow views to be defined separately for the grid and details portions. This means that the view selector for the “grid” and “details” can show different sets of available views. The special casing of view support on these forms also allows the "grid" views to allow filters in their view definitions, whereas the "details" view only need personalizations.

- **More than one modeled form in a single “user-perceived” page**: The ability to embed subforms into modeled forms (via FactBoxes or form parts) leads to situations where more than one modeled form corresponds to a single “user-perceived” page. For example, consider the details portion of the All customers page, which has a number of FactBoxes and two FastTabs whose contents come from form parts. With traditional personalization, contrary to a user’s expectations, exporting the personalizations for the CustTable form would not include personalizations on any FactBox or any personalizations done inside the Addresses or Contact information FastTabs. Equally unexpected for users, any personalizations done on these subforms would also be reflected in other parts of the application where these form parts are used. For example, changes to the Addresses FastTab in the All customers page would also result in the same changes appearing in the Addresses FastTab in the All vendors page.

  With views, the personalization scope of form parts has been modified to match user expectations. In particular, subform personalizations are now tied to the base form where they are made. This means that exporting a view on the All customers details page will include personalizations from the base CustTable form as well as any personalizations in FactBoxes or form parts modeled in that portion of the form. Similarly, any personalizations done on the Addresses FastTab (form part) on the All customers page will not be reflected in other forms where that form part is used.

These are highly technical modifications to the personalization subsystem that are only available when view is enabled. These modifications are important for ensuring that users have a predictable and understandable experience with views.
View support
The style of a form determines the level of support for views.

- Views include queries for forms, such as:
  - List pages
  - Simple lists
  - Grid portions of Master Details and Transaction Details forms

- Views do not include queries, such as:
  - Any other full-page form
  - Details portions of Master Details and Transaction Details forms

- Views that are not currently supported include:
  - Dashboards
  - Workspaces
  - Dialogs
  - Secondary forms like drop-down dialogs, lookups, and enhanced previews

Modifying forms to fully utilize views
While most forms will work well with saved views, there are some areas that may require changes to form logic so that views work as expected on these forms without causing confusion. Here are some key items to keep in mind during development of new forms.

- X++ code late in the form startup cycle can interfere with views working as users expect. In particular, be aware of the following items:
  - Modifications of the query after `super()` of `executeQuery()` or after `super()` of `run()` can cause the query aspect of a view to be ignored.
  - Form changes after `super()` of `run()` can cause some user personalizations to be incorrectly applied to the default view.

- Extra work may be required to ensure the values of custom filters always align to the current view or query.
  - To make sure that custom filters work correctly with saved views, the platform still has additional work to better support these controls. Once that support is available, uptake will be required by any form that has custom filters. More information will be provided when the recommended approach has been finalized.

- Looking forward, in the long-term, views are meant to replace modeled secondary list pages.
  - Typically, secondary list pages, such as Customers on hold, are menu items that point to the same form but have a different query. Because menu items that pass in queries override any query that is defined on the default view, these entry points can cause confusion for users. The current long-term plan is to make secondary list pages obsolete (deprecated) and move them to views. However, that effort hasn’t yet been started.
  - To avoid user confusion between form caption (such as “All customers”) and view name (such as “My customers”), consider renaming form captions to be the name of the corresponding entity. For example, instead of a form caption of “All customers” or “All sales orders”, the form caption would be modified to “Customers” and “Sales orders”.

Known issues
This section provides a list of known issues for saved views while the feature is in a preview state.

**Open issues**
- A view does not get marked as having unsaved changes after using custom filters, which are the filters above a grid excluding the QuickFilter. If custom filter conditions have been saved to a view, the custom filter controls may not correctly reflect the current query.
- View support for workspaces, dashboards, and dialog boxes.
- [KB 4553227] After adding (reference group) fields via personalization, the fields remain blank.
- If the Filter pane is open when switching to a different view, the Filter pane will not update to reflect the filters on the target view.
- Cannot move a view with a QuickFilter condition saved to it to another environment. The fix in release 10.0.13 more gracefully handles the situation, but does not allow these conditions to move between environments.

**Fixed in release 10.0.16**
- [KB 4590240] Grid resize does not work properly when switching views with the old grid
- [KB 4600209] Personalizations of form parts are not reflected when switching views
- [KB 4590224] Focus can start on the wrong control when saved views is enabled
- [KB 4562254] Table permission error after accessing a shared custom workspace
- [KB 4600210] Unexpected client error when switching to the Hide tool
- [KB 4599871] Workspaces do not open if personalization is turned off for user / Unbound controls cannot be set as mandatory via personalization
- [KB 4594453] Duplicate key exception for forms opening as full-page forms and dialogs

**Fixed in release 10.0.15**
- (Quality update) [KB 4599871] Workspaces do not open if personalization is turned off for user / Unbound controls cannot be set as mandatory via personalization
- (Quality update) [KB 4600209] Personalizations of form parts are not reflected when switching views
- [KB 4594452] Duplicate record error when interacting with some subforms (form parts)
- [KB 4586310] Attachments page loses context after switching views
- [Bug 494204] Error when deleting/clearing personalizations from User options > Personalization

**Fixed in release 10.0.14**
- (Quality update) [KB 4594452] Duplicate record error when interacting with some subforms (form parts)
- (Quality update) [KB 4600209] Personalizations of form parts are not reflected when switching views
- (Quality update) [KB 4584077] Error when exporting multiple views
- (Quality update) [KB 4584775] Record position lost when switching between list and details
- [Bug 481290] Error when trying to re-import personalizations to a set of users
- [KB 4582745] Error triggered when importing user views from one environment to another

**Fixed in release 10.0.13**
- (Quality update) [KB 4594452] Duplicate record error when interacting with some subforms (form parts)
- (Quality update) [KB 4600209] Personalizations of form parts are not reflected when switching views
- (Quality update) [KB 4584077] Error when exporting multiple views
- (Quality update) [KB 4582719 and KB 4578126] When multiple personalization records exist for a form, the wrong one can be selected and loaded
- [Bug 481283] Error opening a form after moving a view with a QuickFilter condition between environments
- [Bug 481608] Database sync fails because of a unique index violation on the FormRunConfigurationPublishedView table
- [Bug 474817] User options > Personalization doesn't list all personalizations for the user
- [KB 4574781] Duplicate record exception on saving a view
- [KB 4575278] Tiles, lists, and links lose their link to the published view if the view is republished

**NOTE**
Because additional information is needed to restore the link, re-linking will not occur for any pinned elements from published views prior to 10.0.13. To mitigate, you will need to re-publish your views after updating to 10.0.13 and re-pin the elements to your workspace.

- [KB 4575285] Publishing to an existing view name overwrites configuration changes already made
- [KB 4574778] Pin and publish as default do not respect companies that the view was published to
- [KB 4568154] View import flow doesn’t surface if views apply to the grid or details aspect of Details pages
- [KB 4568152] Users are able to export the Standard view
- [KB 4568151] Published views recipients are not updated after republishing from a different legal entity
- [KB 4562137] Views published to a parent security role are not applied to child roles
- [KB 4564528] QuickFilter default field personalization isn’t working as expected with views

**Fixed in release 10.0.12**
- (Quality update) [KB 4582719] When multiple personalization records exist for a form, the wrong one can be selected and loaded
- [Bug 486275] Strange tooltip behavior for saved views
- [KB 4568122] Unexpected queries applied after enabling views
- [KB 4562152] Migration of personalizations after enabling saved views throws exception in some cases
- [KB 4568121] Default view personalizations not applied for users without personalization rights
- [KB 4568119] Error may occur when importing a workspace or adding a tile or list to a workspace with views enabled
- [KB 4568118] Error when trying to open the view selector when user has a large number of views
- [KB 4568148] Old custom user workspaces aren’t shown in the Personalization form after enabling views

**Fixed in release 10.0.11**
- (Quality update) [KB 4562147] Importing personalizations to a large number of users is timing out
- [KB 4549735] Personalization form missing from security role
- [KB 4568116] Views are not marked as having unsaved changes after using Advanced filter or sort
- [KB 4568117] Crash when attempting to import old personalization formats
- [KB 4568115] Views can be published with no name
- [KB 4564908] Unsaved filters and personalizations reflected in some views

**Fixed in release 10.0.10/Platform update 34**
- (Quality update) [KB 4560406] Importing personalizations to a large number of users is timing out
- (Quality update) [KB 4564906] Personalization form doesn’t load/Loading a published view for the first time takes a long time
- [KB 4568114] Views can be published to a blank legal entity
- [KB 4568113] "View query cannot be applied" message shown when loading a view that modifies existing filters
This topic how to test forms using custom patterns.

Introduction

By adhering to form patterns, you gain various benefits. For example, form patterns correctly set layout properties so that forms are laid out responsively. However, when form pattern coverage is lacking (for example, there currently isn't support for many extensible controls), or when a form or container has unique requirements/uses that don't fit any pattern, developers can set the pattern to Custom. The developer then becomes responsible for ensuring a correct and responsive form layout.

Forms that use custom patterns

You can find the forms that use custom patterns by using the Form Patterns report. For information on running the report, see Form pattern add-ins. After running the report, filter the Percent covered controls column to show forms that have less than 100-percent coverage. For forms that have a top-level Custom pattern, Custom will appear in the Patterns column.

Testing configurations

Key resolution

- 1366 × 768 is a typical resolution on screen sizes that are between 12 and 23 inches. Therefore, this resolution provides a good baseline for testing.
- It's also a good idea to test on a higher resolution, such as 1920 × 1080.

Parameters

- Browser
  - Internet Explorer 11
  - Google Chrome
  - Microsoft Edge
  - Apple Safari (on iPad) – You’ll have to point Safari to a cloud URL.
    - Landscape and portrait modes
- Density
  - Low density
  - High density
- Viewport size
  - Maximized window
  - Snap view (half the screen) to simulate portrait mode on a tablet
- Zoom
  - 50 to 200 percent

Steps

Follow these steps. At each step, examine your form for layout issues. As part of your examination, look at all tab pages, and any groups that can expand/collapse content.

1. Open a browser window at full-screen size, and navigate to your form/control (in low density). It's a good
idea to starting your testing at a resolution of 1366 × 768.

2. Press the Windows logo key+Left Arrow to switch the browser window to Snap view (half the screen).

3. Slowly resize the browser horizontally back to full width. Stop at intervals, and evaluate the form layout.

4. Slowly resize the browser vertically from full height to half-screen height. Stop at intervals, and evaluate the form layout.

5. Maximize the browser window to full-screen size. Adjust the zoom levels (50 percent, 75 percent, 125 percent, 150 percent, and 200 percent), and evaluate the form layout.

6. Do a sanity check in high density.

7. Do a sanity check in other browsers.

Visual issues that you might encounter

**Questions to answer while you're testing**

- Is the form usable?
- Can I reach everything on the form? (You might have to move multiple scrollbars in smaller viewports.)

**Issues that might suggest problems with the layout metadata**

- Form content isn't being adjusted based on the available space.
- Grids and other “fill” controls aren't taking up the full height/width.
- Containers aren't showing up at all (there are no scrollbars that you can use to get to parts of the form).
  - This issue can occur with `SizeToAvailable` containers when there is no available height/width.
- There are extra (unnecessary) scrollbars.
  - You should expect more scrollbars in smaller viewports, especially because some containers (for example, grids and tab pages) have a minimum height.
  - This issue can occur with `SizeToContent` containers that should be `SizeToAvailable`.

**Other known/expected issues**

- Horizontal scrollbars appear on Toolbars.
  - Where there isn’t enough width to show all the buttons on a Toolbar, a horizontal scrollbar will appear.
    - This issue will be resolved soon by a framework deliverable that will implement overflow behavior on Toolbars.
- Random input control borders are missing at various zoom levels.
  - This is a known issue and is tracked by 1721990.
- Grids receive extra scrollbars, because the space that is available for the grid is less than the grid's minimum height (200 px).
  - We have a future work item to investigate reducing/removing this minimum height.
- When StaticText controls that have `SizeToAvailable` width are inside a group that also has `SizeToAvailable` width, horizontal scrollbars appear at some widths (Internet Explorer only).
  - The browser has a calculation error that sometimes causes scrollbars to appear in this scenario.

Appendix

**Information/guidelines about layout**

For information about the layout properties, and for guidelines about scenarios that you should avoid, see Page layout.
Learn how to create shareable, secured URLs to forms and records.

Overview

The URL Generator enables developers to create shareable and secured URLs (also known as deep links) to specific forms that are root navigable. An optional data context can be passed to the form to display filtered or specific data when the form is opened. The URL Generator enables scenarios such as embedding links in reports, email, and external applications, enabling users to quickly and easily locate the specified forms or data by simply navigating using the generated link.

Purpose

- Empower developers to generate URLs that can be used to navigate to root navigable forms in a specified instance.
- Empower developers to optionally specify a data context that should be displayed when navigating to the specified form.
- Empower users to share, save, and access the generated URLs from any browser with Internet access.
- Secure the URLs to prevent unauthorized access to the system, forms, or data.
- Secure the URLs to prevent exposure of sensitive data or tampering.

Security

Site access
Access to the domain/client is controlled through the existing login and SSL mechanism.

Form access
Access to forms is controlled through Menu items, as Menu items are the entry points where security is enforced. If a user navigates using a URL that contains a Menu item that the user does not have access to, then the Menu item security will prevent the form from opening. The user will receive a message indicating that they do not have the necessary permissions to open the form. Note that deep links will only work for Menu items that allow root navigation.

Data access
Access to data is controlled through the existing form-level queries. When a form is opened with a generated URL, the form will run its existing form-level queries, which restrict the user’s access to data. The data context that is specified in the generated URL is consumed after these form-level queries are applied, and results only in further filtering of the data displayed to the user. In short, a generated URL can, at most, open a form and display all of the data that a form would display to the user based on the form-level queries. A generated URL cannot grant a user access to data that is otherwise inaccessible on the form when not using the generated URL.

Usage

The URL Generator is a .NET library that is accessible from X++, under the following namespace.


Requirements
The URL Generator must be used from code running on the AOS, in an active user session or batch process. This requirement ensures that the URL can be secured through encryption specific to the instance that generates the URL. At a minimum, the following information must be specified and passed to the URL Generator in order to generate a working URL.

- **Host URL**
  - The URL of the web root for the instance. For example: https://ax.dynamics.contoso.com/

- **AOT name of the Menu Item Display**
  - The menu item display to be used to open the form.

- **Partition**
  - The partition to use for the request.

- **Company**
  - The company to use for the request.

**Example**

```csharp
// gets the generator instance
var generator = new Microsoft.Dynamics AX.Framework.Utilities.UrlHelper.UrlGenerator();
var currentHost = new System.Uri(UrlUtility::getUrl());
generator.HostUrl = currentHost.GetLeftPart(System.UriPartial::Authority);
generator.Company = curext();
generator.MenuItemName = <menu item name>;
generator.Partition = getCurrentPartition();

// repeat this segment for each datasource to filter
var requestQueryParameterCollection = generator.RequestQueryParameterCollection;
requestQueryParameterCollection.AddRequestQueryParameter(
    <datasource name>,
    <field1>, <value1>,
    <field2>, <value2>,
    <field3>, <value3>,
    <field4>, <value4>,
    <field5>, <value5>
);

System.Uri fullURI = generator.GenerateFullUrl();

// to get the encoded URI, use the following code
fullURI.AbsoluteUri
```
This topic describes best practices for enabling accessibility in your form, product, or control. An accessibility checklist is also included.

Accessibility is about inclusion, that is, a person with a disability can perform the same task as a person without that disability. Making an accessible control or form should be as fundamental as making it secure, high-performing, or easy-to-understand.

### Keyboard

The bedrock of accessibility is keyboard-only access. When you can use the keyboard to perform all the actions of a form, then it can be utilized by a non-sighted person or a person with restricted or limited use of their hands. This means that all controls can be reached via the tab sequence, direct action (such as Ctrl+S for Save), or through some other shortcut key that enables the user to move to a control, such as Navigation Pane, App Bar, Message Center, or Message Bar. A simple test is to simply disconnect your mouse and complete all core and secondary scenarios using only the keyboard.

### Color

The use of color is encouraged and is a common way to express state or status of a record or other piece of information. However, color cannot be the only way that state or status is communicated. An accompanying symbol, help text, or additional column should include a textual description of the state or status. A simple test is to identify all use of color in your system and ensure that color isn’t being used to express a state or status. A common example is to use the color red to indicate “needs attention,” or the color green to mean an “OK” status.

### Images

When showing an image there should be a label that describes the image. If the image expresses state or status of a record, then accompanying help text or an additional column should include a textual description of the state or status. If the image is symbolic, like a logo, then it doesn’t require a textual description. If you have an image on a form or grid to convey a status, such as “in progress”, ensure that the image has a tooltip that can then be read to someone who is utilizing a screen reader.

```java
public display container statusImageDataMethod()
{
    ImageReference statusImage;
    if (this.Status == NoYes::Yes)
    {
        statusImage = ImageReference::constructForSymbol(ImageReferenceSymbol::Accept, "Accept");
        // a product ready example would use a label in place of an embedded string
    }
    else
    {
        statusImage = ImageReference::constructForSymbol(ImageReferenceSymbol::Cancel, "Cancel");
        // a product ready example would use a label in place of an embedded string
    }
    return statusImage.pack();
}
```
Extensible controls

Extensible controls are simply an extension of the controls that are provided by the framework, but require the same usability and accessibility. In addition, widgets or other visually rich controls, such as segmented entry controls or charts, should provide an Accessible Rich Internet Applications (ARIA) tag that is hidden from the sighted user, but provides an introduction to the blind user through a screen reader that describes the use of the control. Every action in an extensible control must be possible with a keyboard.

Layout and dynamic management of forms

A visually impaired or blind user cannot be surprised by significant or unexpected re-lying out of a form based on an action such as selecting a value or entering input.

Accessibility checklist

<table>
<thead>
<tr>
<th>MEASURE</th>
<th>DESCRIPTION</th>
<th>SUCCESS CRITERIA</th>
<th>VALIDATED (X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyboard-only access to all functions</td>
<td>Drill downs, lookups, hyperlinks, data input actions, and shortcut keys.</td>
<td>All action may be completed without the use of a mouse.</td>
<td></td>
</tr>
<tr>
<td>Visible focus indicator</td>
<td></td>
<td>It's always apparent which control has focus.</td>
<td></td>
</tr>
<tr>
<td>Focus order</td>
<td>Cannot “jump to” a non-logical location.</td>
<td>Tabbing doesn’t jump to a non-logical portion or unexpected portion of the form.</td>
<td></td>
</tr>
<tr>
<td>Focus trapping</td>
<td>Cannot be “trapped” in a control, focus must be able to move out of control using the keyboard.</td>
<td>Tabbing into a control should allow tabbing out or a keyboard equivalent that lets them escape.</td>
<td></td>
</tr>
<tr>
<td>Images of text</td>
<td>Cannot use an image to display text.</td>
<td>For example, you cannot have a button label that is an image of text. Logos are exempt.</td>
<td></td>
</tr>
<tr>
<td>Using color</td>
<td>Color alone cannot be used to convey status or state.</td>
<td>All status indicators on a form or grid must have unique shape or texture for each color. See the “Color” section in this document for more information.</td>
<td></td>
</tr>
<tr>
<td>Text contrast</td>
<td>Minimum 4.5:1 ratio.</td>
<td>Extensible controls should use framework color theming to ensure compliance.</td>
<td></td>
</tr>
<tr>
<td>User input instructions</td>
<td>Widgets or other multi-step controls must have a usability overview label or help text (can be embedded in ARIA tag).</td>
<td>When using a screen reader, the control must be introduced and its label or purpose described.</td>
<td></td>
</tr>
<tr>
<td>MEASURE</td>
<td>DESCRIPTION</td>
<td>SUCCESS CRITERIA</td>
<td>VALIDATED</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Change of context</td>
<td>Changing the setting of or entering data into any UI component must not automatically cause an unexpected change of context that might disorient the user without first notifying the user that the change of context will occur.</td>
<td>For example, selecting a value in a drop-down box or clicking a check box shouldn’t change the layout of the form unexpectedly or in a non-standard way.</td>
<td>(X)</td>
</tr>
<tr>
<td>Consistent UI</td>
<td>Widgets or other elements must work consistently across the product. There cannot be two similar looking or modeled elements that behave differently.</td>
<td>For example, selecting a check box shouldn’t open a window.</td>
<td>(X)</td>
</tr>
<tr>
<td>Fine motor control</td>
<td>Cannot require use of the mouse. Must support ‘sticky keys.’</td>
<td>User cannot be forced to click a moving target (pull right) or any other action that requires a mouse. The same action must also be possible with the keyboard.</td>
<td></td>
</tr>
<tr>
<td>Use of video</td>
<td>Video (with audio) must have accompanied text.</td>
<td>When showing pre-recorded content, the deaf person must have supporting text to consume the content.</td>
<td></td>
</tr>
<tr>
<td>Use of images</td>
<td>All images must be accompanied by text that describes the specific content. For example, a “tooltip” or secondary column in a grid that describes and matches the state or status.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication</td>
<td>Use of VoIP or other telecommunication device or software, such as Skype, must be accessible and usable from the keyboard.</td>
<td>Any use of VoIP or telecommunication is a legal obligation.</td>
<td></td>
</tr>
<tr>
<td>Navigation</td>
<td>Navigation controls must communicate that navigation will occur if executed.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Extensible controls – insights for accessibility**

An extensible control is simply an extension of the framework. The interaction of the extensible control should be considered no different than any other framework control. When the control is a visually-rich widget that offers mouse actions, the author needs to ensure that equivalent, keyboard access functionality is available. A widget may not run in an “accessible mode” that is different than the standard presentation. Instead the widget
must offer similar functionality without the need to activate or toggle state of the control. All uses of color should be based on themed colors so that when the mode is “High Contrast” your color scheme matches the theme change. The World Wide Web Consortium (WC3) website provides guidance for Supported States and Properties. This site is a helpful resource for ARIA tags and provides the definitions that you see below.

**Controls should do this**

**Introduce itself**

Importantly, your control should not only identify itself by name, but (using a label or ARIA tag) give a brief introduction on how it works. Make sure your descriptive text is supplied via a framework label for localization.

- *aria-describedby* - Identifies the element (or elements) that describes the object.

**Introduce itself - example**

```html
<button aria-label="Close" aria-describedby="descriptionClose" onclick="myDialog.close()"></button>

<div id="descriptionClose">Closing this window will discard any information entered and return you to the main page</div>
```

**Indicate when it is busy**

It may not always be clear to the visually-impaired user why the control isn’t responsive. Providing a “busy” message helps in these cases.

- *aria-busy (state)* - Indicates whether an element, and its subtree, are currently being updated.

**Indicate when it is busy - example**

```html
<p aria-live="polite" aria-busy="true"></p>
```

**Indicate that the contents have been validated and are invalid**

The async nature will result in a dynamic field state change. The message bar will introduce itself to the visually-impaired user, and the control itself should express an invalid state.

- *aria-invalid (state)* - Indicates the entered value does not conform to the format expected by the application.

**Indicate when a field is readonly**

- *aria-readonly* - Indicates that the element is not editable, but is otherwise operable. See related *aria-disabled*.

**Indicate that the field requires input (mandatory)**

The sighted user understands that a field is mandatory through a visual symbol. The non-sighted user will need an identifying tag.

- *aria-required* - Indicates that user input is required on the element before a form may be submitted.

**Describe the state of a toggled value**

A toggle control has a toggled state. This tag will express that state.

- *aria-pressed (state)* - Indicates the current “pressed” state of toggle buttons. See related *aria-checked* and *aria-selected*.

- *aria-valuenow* - Defines the current value for a range widget. See related *aria-valuetext*.

- *aria-valuetext* - Defines the human readable text alternative of *aria-valuenow* for a range widget.

**Controls could do this**

**Indicate an expanded state**

Complex interactions can be learned, but current state isn’t always easy to determine without experimentation. When using an *aria-expanded* tag, the control describes its current state. An example is tabbing to a tab or FastTab section of a control.
Describe applicable context menu

Finance and Operations apps provide a context menu. When the application author has provided functionality to the current control or context, you can announce that functionality.

- `aria-haspopup` - Indicates that the element has a pop-up context menu or sub-level menu.

Other miscellaneous controls

- `aria-live` - Indicates that an element will be updated, and describes the types of updates the user agents, assistive technologies, and user can expect from the live region.
- `aria-multiline` - Indicates whether a text box accepts multiple lines of input or only a single line.
- `aria-multiselectable` - Indicates that the user may select more than one item from the current selectable descendants.
- `aria-orientation` - Indicates whether the element and orientation is horizontal or vertical.
- `aria-setsize` - Defines the number of items in the current set of listitems or treeitems. Not required if all elements in the set are present in the DOM. See related `aria-posinset`.
- `aria-sort` - Indicates if items in a table or grid are sorted in ascending or descending order.
- `aria-valuemax` - Defines the maximum allowed value for a range widget.
- `aria-valuemin` - Defines the minimum allowed value for a range widget.
This article describes how you can customize existing field descriptions and add your own descriptions.

There are descriptions for some of the more complex fields. These descriptions appear when you hover over a field. You can customize these descriptions if, for example, you want to add company-specific information. You can also add descriptions for additional fields. You create field descriptions by using the **HelpText** property for field controls. The **HelpText** property is no longer specified for table fields and data types, as it was in previous versions. Additionally, the inheritance of the **HelpText** property from data types and table fields to form controls is obsolete. Field descriptions are intended to be specific to an individual field, in the context of the other controls and information that are available on the page. To add and customize field descriptions, you must have access to the development environment. Like other metadata changes, new descriptions should be added in a new model to prevent them from being overwritten when a new version of Operations is released. For more information, see Customizing through extension and overlayering.

### Customize a field description or add a new description

The same procedure is used to customize existing field descriptions and to add new field descriptions. However, when you customize an existing description, you replace the existing label reference.

1. In Application Explorer, find the relevant page (form), and add it to your project.
2. In the node for the page, find the relevant field control. Make a note of the name, so that you can use it as part of the label ID.
3. Add a new label for your description. You can follow the conventions that Microsoft uses to name the label files for field descriptions and to create the label file IDs. For more information, see the next section.
4. In the **HelpText** property of the field control, add a reference to the label.

### Label file names and label IDs

The field descriptions that are provided by Microsoft are stored in separate label files. There is one label file per module per model. The pattern for the label file names is `FieldDescriptions_ModuleName_ModelName.CountryCode.label.txt`. Here are some examples:

- FieldDescriptions_AccountsPayable_ApplicationFoundation.en-US.label.txt
- FieldDescriptions_SystemAdministration_ApplicationFoundation_en-US.txt

The pattern for label IDs is `@FieldDescriptions\_*ModuleName:PageName*\_*ControlName*`. Here are some examples:

- @FieldDescriptions\_AccountsPayable:CustSettlement\_CustSettlement\_OffsetCompany is the ID for the label for the **Company accounts** field (CustSettlement_OffsetCompany) on the **Customer settlement** page (CustSettlement).
- @FieldDescriptions\_ProcurementAndSourcing:PurchLineBackOrder\_LinkViewCheckBox is the ID for the label for the **Link on change view** option (LinkViewCheckBox) on the **Backorder purchase lines** page (PurchLineBackOrder).

### Additional resources

Create localizable labels
Actions are an essential component of any enterprise resource planning (ERP) system, and are triggered by mouse click, keyboard, or touch.

Introduction

Actions are an essential component of any enterprise resource planning (ERP) system. Actions can be accessed through various mechanisms:

- Buttons on standard Action Panes
- Buttons on Toolbars
- Buttons directly on the form canvas
- Right-click context menus
- Keyboard shortcuts
- The new action search feature

Note that, in general, actions that are triggered by right-click context menus or keyboard shortcuts are meant to have a corresponding button available elsewhere in the user interface. Action controls can be triggered by using touch or a mouse click. Many system-provided actions can also be triggered by using the keyboard. In the future, we plan to provide functionality so that developers and end users can also define their own keyboard shortcuts.

Buttons

Buttons are the foundation of action controls. They can be modeled inside of standard Action Panes or in Toolbars, which are discussed later in the article. They can also be added as stand-alone buttons on the page (for example, the OK and Cancel buttons at the bottom of a dialog box, or buttons for actions that are specific to an individual field). The following button types continue to be available:

- A button is a basic button, for which the entire functionality must be implemented in code.
- A command button specifies a command or task to run.
- A menu item button specifies a menu item to navigate to or run.
- A drop dialog button opens a flyout dialog box, the contents of which are retrieved via a menu item.
- A menu button is a button container that opens a flyout that contains a list of other buttons.

In general, buttons continue to take advantage of the same properties as the buttons in previous versions. The following sections discuss a few properties that are related to button visualization, with particular focus on changes from previous versions.

Button Display

The Button Display property controls what information (including the button label and/or image) appears on the button. The allowed values for this property depend on where the button is located (for example, inside an Action Pane). Here are some of the location restrictions on the Button Display property:

- Buttons inside Menu Buttons must be set to Text Only or Auto (which is interpreted as Text Only in this case).
- Buttons inside Action Pane tabs on Standard Action Panes must be set to Text Only or Auto (which is interpreted as Text Only in this case).
- **Image Only** buttons should be used only for on-canvas buttons that are inline with a field.

For more details about how to use the **Button Display** property in various form locations, see the “Button image guidelines” section of the *General form guidelines* article. The following table shows the values for the **Button Display** property.

<table>
<thead>
<tr>
<th>BUTTON DISPLAY VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto</td>
<td>The button has the content (text and/or image) that is defined (but only content that is allowed for the button’s location).</td>
</tr>
<tr>
<td>Text Only</td>
<td>Only text appears on the button.</td>
</tr>
<tr>
<td>Image Only</td>
<td>Only an image appears on the button.</td>
</tr>
<tr>
<td>Text with Image Above</td>
<td>Both an image and text appear on the button. The text appears above the image.</td>
</tr>
</tbody>
</table>

Note that the other values of **Button Display** from Microsoft Dynamics AX 2012, such as **Background Image** and other relative positioning of text and image, are no longer supported.

**Button Images**

In previous versions, images or icons were often shown on buttons to help users recognize those buttons. In the current version, the number of images that are used for this purpose has been drastically reduced. Fewer images produce a cleaner, more modern user interface. Additionally, there was a desire to indicate processes and tasks by using more common symbols instead of multiple subtly different images. For more details about how images are used on buttons, see the “Button image guidelines” section of the *General form guidelines* article.

Two metadata properties are used to define an image for a button: **Image Location** and **Normal Image**. The allowed values for the **Normal Image** property depend on the value of the **Image Location** property. In previous versions, Embedded Resources (kernel resources) were heavily used to specify button images or icons. However, with the shift to the web, this image format option is no longer available. Instead, a new image format (Symbol font) has been added, and the expectation is that all buttons that require images will use this format (**Image Location = Symbol**). The primary reason for this change is that a symbol font is the best performing and most scalable image format. For a list of the full set of symbols that are supported, see *Dynamics Symbol font* article. The following table shows the recommended and preferred method for assigning images to buttons.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Location</td>
<td>Symbol</td>
</tr>
<tr>
<td>Normal Image</td>
<td>The name of the symbol font glyph</td>
</tr>
</tbody>
</table>

**Button Style**

In general, the **Button Style** property defines how a button is shown in the user interface. The exceptions are buttons that are modeled inside an Action Pane or Toolbar, because the **Button Style** property is disregarded in those cases. Instead, those buttons are rendered by using the style specially designated for buttons in those types of containers. For buttons that are modeled directly on the form canvas (outside Action Panes), the following button styles are available.
### Standard Action Panes

The standard Action Pane is the primary location for page-level actions. It consists of both system-defined actions (actions that aren’t explicitly modeled but are automatically added by the framework) and developer-defined actions (actions that are explicitly modeled in either Action Pane tabs or Button Groups). Developers can promote the most frequently used actions directly to the standard Action Pane by modeling Button Groups directly under the Action Pane. However, Action Pane tabs can still be used to group actions and provide access via a flyout. The following illustration shows a standard Action Pane that includes system-defined buttons, two promoted developer-defined actions, and a set of Action Pane tabs.

The following illustration shows the flyout that appears to show additional commands when an Action Pane tab is clicked.

### System-defined buttons

Several system-defined buttons are added automatically to pages. The following table shows the list of system-defined buttons that are added to the Action Pane. For more information about how these buttons behave and how to manage them, see the System-defined buttons article.

<table>
<thead>
<tr>
<th>BUTTON</th>
<th>NAME</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="New" /></td>
<td>New</td>
<td>Create a new record for the first master data source.</td>
</tr>
<tr>
<td><img src="image" alt="Delete" /></td>
<td>Delete</td>
<td>Delete the currently selected record for the first master data source.</td>
</tr>
<tr>
<td><img src="image" alt="Edit" /></td>
<td>Edit</td>
<td>Switch to Edit mode.</td>
</tr>
<tr>
<td><img src="image" alt="Show filters" /></td>
<td>Show filters</td>
<td>Open the Filter pane.</td>
</tr>
</tbody>
</table>
### Pinning the Action Pane

The standard Action Pane supports the ability for the user to "pin" or "unpin" the Action Pane as desired.

When the Action Pane is pinned open, an Action Pane tab is expanded and pushes the form content below it (e.g. it does not overlap anything on the form). In this mode, there is a chevron button in the lower right corner of the expanded Action Pane tab to "unpin" the Action Pane.

When the Action Pane is not pinned open, clicking on an Action Pane tab opens it as a flyout on top of the form content. The lower right corner of the Action Pane tab flyout has a pushpin button that can be clicked to pin the Action Pane open.

### Overflow behavior in the Action Pane

Standard Action Panes include an overflow feature that adds a responsive element to forms and eliminates the need for a horizontal scrollbar in the Action Pane. When the browser width is insufficient to show the entire Action Pane contents, an overflow menu automatically appears in the Action Pane and includes any buttons and
Toolbars

Toolbars (previously called Action Pane strips) are Actions Panes that have the **Style** property set to **Strip**. They are used for actions that have a specific context and aren’t page-level actions. They are primarily used for actions that are specific to a FastTab, tab, or grid. The actions in a Strip-styled Action Pane are shown horizontally in a Toolbar. The following illustration shows a Toolbar that has two buttons for adding and removing lines from this **TransactionDetails** form.

### Overflow behavior in Toolbars

Toolbars have the same overflow feature as standard Action Panes. See the section above for more details.

Toolbars allow developers to provide a cleaner action story by designating certain buttons to always render in overflow. This makes it easier to differentiate actions that users will commonly use versus those that are infrequently or rarely used. This behavior is controlled by a new metadata property called **AlwaysInOverflow** that exists on Button groups inside Toolbars.

### Right-click context menus

Some actions can also be accessed via shortcut menus (right-click context menus). Depending on the context of the right-click, you see either the browser’s default context menu or the application context menu, which shows both system-defined actions and developer-defined actions.

- If you right-click on an image, in an editable field, or if text is selected, then the browser’s context menu will appear. This is to provide access to browser functionality like **Cut**, **Copy**, and **Paste**. Actions cannot be embedded into context menus because browsers do not allow programmatic access to the system.
The programming model for modifying context menus differs from the model used in previous releases. In Dynamics AX 2012, the `PopupMenu` class was used. This class relies on Microsoft Windows application programming interfaces (APIs). However, because these APIs aren't available on the web, replacement APIs have been created to provide similar functionality. For more information, see Code migration - Context menu code.

### Keyboard shortcuts

Keyboard shortcuts are another mechanism for triggering some actions. Many actions that had shortcuts in Dynamics AX 2012 continue to have shortcuts in Operations. However, because of browser restrictions, the key combination that is used to trigger a particular action might differ.

The following table shows some important keyboard shortcuts that are available. For the full list of current keyboard shortcuts, see the Keyboard shortcuts article. In the future, we plan to provide mechanisms so that developers and end users can define shortcuts for other actions.

<table>
<thead>
<tr>
<th>KEY COMBINATION</th>
<th>ACTION</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alt+N</td>
<td>Create a new record</td>
<td>The keyboard shortcut for creating a new record (unlike the system-defined <strong>New</strong> button on the Action Pane) is contextual. It creates a new record that is based on the data source of the control that currently has the focus.</td>
</tr>
<tr>
<td>Alt+Del (or Alt+F9)</td>
<td>Delete record</td>
<td>The keyboard shortcut for deleting a record (unlike the system-defined <strong>Delete</strong> button on the Action Pane) is contextual. It deletes based on the data source of the control that currently has the focus.</td>
</tr>
<tr>
<td>Alt+S (or Ctrl+S)</td>
<td>Save a record</td>
<td></td>
</tr>
<tr>
<td>F2</td>
<td>Toggle edit mode</td>
<td>Switch the form between <strong>View</strong> mode and <strong>Edit</strong> mode.</td>
</tr>
</tbody>
</table>

### Action search

Dynamics AX 2012 included a Key Tips feature that let users run any command in an action pane by pressing Alt and then a series of letters. The action search feature has been implemented as the replacement for the old Key Tips functionality. Action search can be accessed through a search field that is located in the standard action pane at the top of the form. Currently, it's represented by a magnifying glass icon in the action pane (note this is different from the navigation search feature in the navigation bar). In the action search field, you can type the beginning of the name of the action that you want to perform in the field (typically, only two to four characters
are required). This search mechanism then finds all buttons in visible action panes on the form that match the search string. You can then use click the button in the result list to run the command. For productivity, focus then returns to your last position in the form after the button has been triggered. You can also initiate action search by pressing Ctrl+' or Alt+Q. Pressing the keyboard shortcut again will return focus to your last position in the form.
This topic describes how to create a consistent look and feel for forms by controlling the size of controls and grids.

Overview

Many frameworks offer complete freedom over the width of input controls. However, that level of freedom can lead to inconsistent presentation of data and a non-uniform layout for similar forms. For example, the customer name in one form might show 10 characters of information, whereas the customer name in another form might show 20 characters of information. To provide streamlined, easy-to-read interfaces, discrete sizing helps guarantee consistent presentation of data. The introduction of discrete sizing is a significant change to the basic input controls. The control framework attempts to provide a fresh, clean user experience that provides simplicity and consistency. As part of an attempt to provide consistent and uniform layout of forms, each input control is sized to one of four sizes: extra-small (XS), small (S), medium (M), or large (L). These sizes are determined by inspecting the explicitly specified width in the `DisplayLength` property of the control or the corresponding extended data type (EDT).

Sizing grid columns

Column sizing in a grid control differs slightly from the legacy algorithm, which tried to provide an initial appealing presentation for each grid, based partly on the contents of the first page of data. The new approach disregards the contents of the first page of data. Instead, the end user decides which columns are most important to view and the ideal viewing width for each of those columns. By using personalization, each user can change the width of grid columns, and the client keeps track of that user's preference. In the new, simplified approach, the default column sizing is based on 75 percent of the base input control sizing. In most cases, we recommended that developers not override the default sizing. However, if you must resize the column width programmatically or in the model, see the next two sections of this article for guidance.

### Discrete sizing

<table>
<thead>
<tr>
<th>SIZE</th>
<th>CHARACTER RANGE</th>
<th>SIZE IN PIXELS (PX)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XS</td>
<td>0–5</td>
<td>60</td>
</tr>
<tr>
<td>S</td>
<td>6–15</td>
<td>120</td>
</tr>
<tr>
<td>M</td>
<td>16–30</td>
<td>180</td>
</tr>
<tr>
<td>L</td>
<td>&gt;30</td>
<td>240</td>
</tr>
</tbody>
</table>

**Note:** The explicit number of pixels will likely vary over time as the user interface evolves. Developers should not rely on explicit pixel sizing.

Forcing a desired discrete size

As the previous table shows, if you change the width of a grid-hosted control from 6 characters to 15 characters, you don't affect the width of the column. The layout engine gives the same width to all control widths that are in
the 6-to-15-character range, because controls in this range are considered small (S). If you want to extend the control to medium (M) size, the width value must be set to a value that is more than 16 characters and less than 31 characters. For guidance about how to size input controls, see the following table. In some cases, the use of a form pattern will override or hide developer-defined sizing.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>DisplayLengthMode</td>
<td>Auto</td>
<td>Use the kernel's default value.</td>
</tr>
<tr>
<td>DisplayLengthMode</td>
<td>Fixed</td>
<td>Use the developer-defined DisplayLength value.</td>
</tr>
<tr>
<td>WidthMode</td>
<td>Auto</td>
<td>Use the kernel's default sizing behavior.</td>
</tr>
<tr>
<td>WidthMode</td>
<td>Fixed</td>
<td>Use the developer-defined Width value.</td>
</tr>
<tr>
<td>WidthMode</td>
<td>Column Width</td>
<td>Use size-to-parent behavior.</td>
</tr>
<tr>
<td>DisplayLength</td>
<td>N</td>
<td>The developer-defined control width, which is specified in characters. (Values are mapped to the sizing groups that are listed in the previous table.)</td>
</tr>
<tr>
<td>Width</td>
<td>N</td>
<td>The developer-defined control width, which is specified in pixels.</td>
</tr>
</tbody>
</table>
This article is intended as a primer for using check box controls in the tree control. It's not a general “how to” for using tree controls.

Microsoft Dynamics AX 2012 includes several examples of tree controls that were enhanced so that they both show data in a tree hierarchy and let the user select one or more nodes by using check boxes. In Dynamics AX 2012, the tree control had no built-in support for check box controls. Instead, an image of a check box was added for each node in the tree control. The image state for each node was then toggled as the user clicked the check box.

The current version has greatly simplified the experience for the developer. Check box support is now built into the tree control.

You no longer have to use images to include a check box, and you also don’t have to explicitly set the state of the check box state when it's selected. The control doesn’t use images, and the check box state is managed in the way that you would expect for a tri-state check box. Examples of tri-state check boxes can be found in most installation scenarios. When tri-state check boxes are used, if the user selects a parent node, all children of that parent also become selected. The check box interaction is independent of the node’s expand/collapse functionality. When the parent node is collapsed (no children are visible), a check mark on the parent node indicates that all children are also selected. However, if one child of a parent that has multiple children isn’t selected, the appearance of the parent node changes. The check box no longer contains a check mark but is filled in. This state is considered a partial check. Therefore, a parent node has three states:
• Checked
• Unchecked
• Partial

If the user clicks the check box on a parent node that is in a partial state, the state of the parent and all its children changes to checked. (The parent node and all its child nodes are now selected.)

Parent node in a partial state

- Element 3
  - Child
  - Child

Parent node and all child nodes in a checked state after the parent node is selected

- Element 3
  - Child
  - Child

If the user clicks the check box on a parent node that is in a checked state, the state of the parent and all its children changes to unchecked. (The parent node and all its child nodes are now cleared.)

Parent node in a checked state

- Element 3
  - Child
  - Child

Parent node and all child nodes in an unchecked state after the parent node is cleared

- Element 3
  - Child
  - Child

If the user clicks the check box on a parent node that is in an unchecked state, the state of the parent and all its children changes to checked. (The parent node and all its child nodes are now selected.)

Parent node in an unchecked state

- Element 3
  - Child
  - Child

Parent node and all child nodes in a checked state after the parent node is selected

- Element 3
  - Child
  - Child

A child node that has no children (in other words, a child node that isn’t a parent itself) has only two states: checked and unchecked. A child node that is the only child in a checked state affects the state of its parent. If a child node is selected, the state of its parent changes to partial. Note: A single node in a tree also has a
Tree controls that contain check boxes in Dynamics AX 2012

The following example is from SysConfiguration.

1. The program checks for the `mouseDown` event.
2. When the `mouseDown` event is detected, the program determines whether the user clicked the node or the image.
3. If the user clicked the image, the program toggles the image state.

None of this code is required for the current version.

```java
int mouseDown(int x, int y, int button, boolean ctrl, boolean shift) {
    int idx, f;
    FormTreeItem parentNode, node;
    int parentMode;
    boolean enabled;

    #FormTreeControl;
    [idx, f] = this.hitTest(x, y);
    parentNode = this.getItem(this.getParent(idx));
    node = this.getItem(idx);
    if (node) {
        if (parentNode) {
            if (element.enabled(parentNode.data()))
                parentMode = true;
        } else
            parentMode = true;
        if ((f & #FTCHT_ONITEMICON) && parentMode) {
            if (!node.overlayImage()) {
                enabled = (element.enabled(this.getItem(idx).data()) ? false : true);
                element.enabled(this.getItem(idx).data(), enabled);
                element.drawTree();
            }
        }
    }
    return super(x, y, button, ctrl, shift);
}
```

In the current version, you still set the selected state for scenarios where the user is presented with preselected nodes. Additionally, the developer can still set the state explicitly when the FormTreeItem is created. However, instead of specifying the current image, the developer now sets the `stateChecked` property on the FormTreeItem. If developers must know when the state of a check box changes, they can override the `checkedStateChanged()` method.

Basic check box use for tree controls in the current version

Make sure that the `Check Box` property on the modeled tree control is set to `Yes`. To explicitly set the state on a node, use the following code.

```java
formTreeItem.stateChecked(FormTreeCheckedState::Checked);
formTreeControl.setItem(formTreeItem);
```
To interrogate a node for its current state, use the following code.

```java
FormTreeItem formTreeItem = formTreeControl.getItem(formTreeControl.getSelection());
FormTreeCheckedState currentState;
if (formTreeItem != null)
{
    currentState = formTreeItem.stateChecked();
    switch (currentState)
    {
        case FormTreeCheckedState::Unchecked:
            /* unchecked */
            break;
        case FormTreeCheckedState::Checked:
            /* checked */
            break;
        case FormTreeCheckedState::Partial:
            /* parent has children checked */
            break;
        default:
            /* shouldn't get here */
            break;
    }
}
```

To react to or track the checked state of a node (idx is the node index), use the following code.

```java
public void checkedStateChanged(int _Idx, FormTreeCheckedState _newState)
{
    super(_Idx, _newState);
}
```
This topic explains the filtering options that are available.

Introduction

Microsoft Dynamics AX 2012 offers the following filtering options.

<table>
<thead>
<tr>
<th>FILTER OPTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter by grid</td>
<td>The user defines filter conditions in input fields below the grid column headers.</td>
</tr>
<tr>
<td>Filter by selection (filter by field)</td>
<td>The user selects a field value and uses that value as a filter condition.</td>
</tr>
<tr>
<td>Advanced filter</td>
<td>The user opens a dialog box that contains advanced filtering options (filter on columns, not on the form; join additional data sources; sort by multiple columns; and so on).</td>
</tr>
</tbody>
</table>

Finance and Operations offers the following filtering options.

<table>
<thead>
<tr>
<th>FILTER OPTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter Pane</td>
<td>An inline pane that slides in from the left, and that contains multiple filter criteria that can be applied to the targeted content.</td>
</tr>
<tr>
<td>QuickFilter</td>
<td>A framework-provided filtering mechanism that can appear above any list or grid, and that provides fast single-column filtering.</td>
</tr>
<tr>
<td>Grid column filtering</td>
<td>The user can define filter conditions and perform single-column sorting by using a drop dialog that is opened from the grid column header.</td>
</tr>
<tr>
<td>Advanced filter/sort</td>
<td>For most advanced filtering scenarios, the migrated Advanced filter form from Dynamics AX 2012 is still available.</td>
</tr>
</tbody>
</table>

Filter expressions

One important difference between filtering in Finance and Operations apps and filtering in Dynamics AX 2012 is related to the way that query symbols are used when filter values are defined (for example, "*" to match 0 or more characters, or ".." to specify a range of values to match). In Dynamics AX 2012, these symbols are highly visible during the filtering experience. For example, for the filter by grid option, if a user selects the contains operator on a field, the system translates that operator by adding wildcard characters (*) to each end of the current expression. In the current version, the query symbols are implied by the selected operator and aren’t injected into the user interface. This makes filtering more intuitive and simpler for users. For users who want to specify additional filter conditions by using specific query symbols, or users who must enter more complex
conditions, the matches operator is provided for each data type. For all other operators, the query symbols are interpreted as literals. For example, the filter condition “First name MATCHES A” finds all records where the first name starts with the letter A. However, the filter condition “First Name IS A*” finds records where the first name is literally equal to “A*.” The following table shows how the client translates between Finance and Operations apps filter operators and Dynamics AX 2012 query syntax.

<table>
<thead>
<tr>
<th>FILTER OPERATOR</th>
<th>FINANCE AND OPERATIONS APPS QUERY SYNTAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is exactly “circle” / Is equal to “circle”</td>
<td>“circle”</td>
</tr>
<tr>
<td>Is not “circle” / Is not equal to “circle”</td>
<td>“!circle”</td>
</tr>
<tr>
<td>Is one of “circle”, “square”, “circlesquare”</td>
<td>“circle,square,circlesquare”</td>
</tr>
<tr>
<td>Contains “circle”</td>
<td>“<em>circle</em>”</td>
</tr>
<tr>
<td>Does not contain “circle”</td>
<td>“!<em>circle</em>”</td>
</tr>
<tr>
<td>Begins with “circle”</td>
<td>“circle*”</td>
</tr>
<tr>
<td>After “circle” / Greater than “circle”</td>
<td>“&gt;circle”</td>
</tr>
<tr>
<td>Greater than or equal “circle”</td>
<td>“circle..”</td>
</tr>
<tr>
<td>Before “circle” / Less than “circle”</td>
<td>“&lt;circle”</td>
</tr>
<tr>
<td>Less than or equal “circle”</td>
<td>“..circle”</td>
</tr>
<tr>
<td>Between “square” and “circle”</td>
<td>“square..circle”</td>
</tr>
</tbody>
</table>

Any query syntax that doesn’t match the preceding templates is interpreted as the matches operator.

**Other frequent filter expressions**

Users may want to filter for blank values in column. While there is no filter operator for this specifically, the syntax for performing this filtering remains the same as Dynamics AX 2012. With either the matches operator or the is equal to operator, users can type “” to retrieve rows with blank values for the current column. For example, First Name IS “” will find all records where the first name is blank. Note that “” only matches rows where the column value is the empty string and does not match rows where the column value is NULL or 0.

Users may also want to filter for records that do not belong to a specific list of values. While “is not one of” is not a filter operator, this filter expression can be achieved using the matches operator by negating each list item in the list. For example, !circle, !square will find all records that are neither “circle” nor “square.”

**Filter Pane**

The Filter Pane provides an easy-to-use interface for filtering full page lists. The Filter Pane is an inline pane that slides in from the left side of the screen and pushes the page content to the right, so that users can see the data that they want to filter. Users open this filter mechanism by clicking the system-defined Show filters button on the left side of the page. After it has been opened, the Filter Pane remains visible until the user goes to a new page, or until the user closes the Filter Pane by using the Hide filters button.

**When is the Filter Pane available?**

Currently, the Filter Pane is available for all forms except the following forms:
What data does the Filter Pane work on?
Because the Filter Pane is targeted at full page lists, it works only on the tables and fields that are directly joined (by inner/outer joins) to the first master data source on the form. This filtering mechanism isn’t intended for filtering on secondary collections, or for filtering on other root data sources and their directly joined data sources. Other filtering mechanisms (QuickFilter, grid column filtering, and so on) are available to meet these other requirements.

What fields are initially shown in the Filter Pane?
Here is how the fields that are initially shown in the Filter Pane are selected:

1. All ranges/filters that currently exist on the query (only non-hidden filters/ranges are shown) are used.
2. If no ranges filters currently exist on the query, the fields from the primary index from the first master data source are used.
3. If there are no fields from the primary index from the first master data source, the TitleFields that are defined directly on the first master data source are used. If no TitleFields are defined, no default fields are shown. (Currently, if the first master data source extends another table (for example, table B), we don't show the TitleFields from table B. However, we plan to add that check in the future.)

Can I control the default fields that appear in the Filter Pane?
Developers can make sure that a particular field appears in the Filter Pane by adding an empty filter for that field to the query. For an example, see the FmCustomer form, which adds the filters post super() in form init(). Note that after an empty field has been added to guarantee that it appears in the Filter Pane, the fields in the Filter Pane will always be those that are explicitly on the query, and will never be the TitleFields or fields from the primary index on the first master data source.

I don't want users to be able to filter on a specific field or modify an existing filter. How do I accomplish this?
Developers can affect whether users can modify/add filters on certain fields by changing the status of the filters. The allowed values are in the RangeStatus enum:

1. Open (default) – The user can see and modify this filter.
2. Locked – The user can see the filter value but can’t modify it. The user also can’t add another filter on this column.
3. Hidden – The user can't see that there is a filter on this column. The user also can't add another filter on this column.

Can I control the fields that appear in the Add a filter field list in the Filter Pane?
The fields that appear in the Add a filter field list are all the filterable fields from the query that involves the first master data source on the form. Therefore, developers can't control the fields that appear in this list. Usually, if you see unexpected fields or can't find the fields that you want to filter on, the fields that you're expecting are either on a different master data source (not the first) or on a child collection.

How is the Filter Pane used?

Note: The availability of the Filter Pane on particular forms and form types is evolving, so this list might change.
The Filter Pane is simple and straightforward to use. First, select a filtering operator in the list that is associated with each filter field. Note that the set of operators that appears depends on the data type of the field. Then enter an appropriate value for the filter condition, and click **Apply**. The form is updated based on the filter criteria that you specified.

**QuickFilter**

In Dynamics AX 2012, the QuickFilter was a framework control that was automatically added only to list pages. In Finance and Operations apps, the QuickFilter is now a modeled control that can be associated with any grid in the system. As the user starts to type, a column selector drop-down appears to guide the user toward the column that the filter will be applied to. The developer can also specify the default column for the QuickFilter. If no column is specified by the developer, the default column is the first field that can be filtered in the grid.

Column selectors are shown only for QuickFilters that are attached to grids. If you don't see a column selector, the most likely reason is that the **TargetControl** property on the QuickFilter is blank. This property must point to the grid that it should operate on. If the **TargetControl** property is set correctly, but you don't see a column selector, you might not have any filterable columns in your grid. In addition to non-text controls (such as images), controls that are bound to data methods aren't filterable.

**Can I use the QuickFilter to filter other collection controls (such as trees)?**

Yes, you can use the QuickFilter to filter other collection controls, but you must manually wire up the filtering. Here are the general steps:

- Leave the **TargetControl** property blank.
- Override the **applyFilter()** method on the QuickFilter.
- Write code in that method to perform the desired filtering.

**Grid column header filtering/sorting**

In Finance and Operations apps, the grid filtering experience is more closely aligned with the experience in Microsoft Excel. When the user clicks a column header (for columns that can be filtered), a drop dialog appears, and the user can use it to filter the column. The filtering experience here mimics the filtering experience in the Filter Pane. Additionally, there are options to sort the grid based on the column that is currently selected.
In Microsoft Power Apps, you can manage organizational data through apps that you created, or apps that someone else created and shared with you. Apps run on mobile devices such as phones or in a browser. Apps can also be embedded in Finance and Operations apps by developers using the Microsoft Visual Studio developer experience. To learn more about Power Apps, see https://powerapps.microsoft.com.

Host an app from Power Apps on a page

1. In Power Apps, find the web-based app that you want to host, and record or copy the App ID value.

2. In Visual Studio, open your project, and then, in the form designer, add an instance of a Power Apps Host control to your page.

3. In the Properties pane, enter the App ID value.

4. If your app shares or is linked to the current data source on your page, you can pass the ID of the primary or linked key field for the data that you want your app to show. In this case, provide the ID as the value of the Entity ID, Entity ID Data Source/Field, or DataMethod property. This value will then be passed to your app as a parameter value, and your app must use that value to obtain the linked data.
5. In some cases, your app might be hosted in a development or sandbox Power Apps environment that is provided by Microsoft. In this case, you must supply that override URL as the value of the **Power Apps Environment Override** property.

Sizing is determined by the container that you put your control in. If you put your control in a form pattern that has limited available space, and your app has been designed to be larger than the available space, your embedded app will have scroll bars.
A programming model is required for context menus (shortcut menus). This topic outlines the process for migrating context menu code from Microsoft Dynamics AX 2012 to Finance and Operations. It also includes user experience (UX) guidelines for context menus.

In Dynamics AX 2012 and earlier versions, developers modified right-click context menus (shortcut menus) by using the `PopupMenu` class. This class relied on Microsoft Windows application programming interfaces (APIs) that aren't available on the web. In Finance and Operations, the `ContextMenu` APIs have been created as replacements to provide similar functionality. Previously, the `context()` and `showContextMenu()` method overrides were the entry points for modifying context menus for specific controls. These overrides typically contained code to add options to the context menu, and also to process the user’s selection. The code for processing the user’s selection used a wait model. Because these overrides are being removed and the wait model is being eliminated, developers must now create two overrides: `getContextMenuOptions()` to add options to the context menu and `selectedMenuOption()` to process the user’s selection.

Migrate context menu code

Migration from the `PopupMenu` APIs to the `ContextMenu` APIs can be broken down into three main steps.

**Step 1. Add a constant for each menu option that must be added**

The old `insertItem()` method in the `PopupMenu` class returned an identifier for the menu option that was being added. This identifier was saved into a variable for future reference. Because developers will define the menu identifier, it's a good idea to define constants for each option to help with code readability.

- At the form level, add a constant for each menu option that is being added to the context menu. The value must be unique within each context menu. Note that you must modify the old variable name if it conflicts with another variable on the form or control.

**Before**

```java
public void context() {
    ...
    int listCreateRoot = listMenu.insertItem("@SYS5480");
    ...
}
```

**After**

```java
[Form]
public class MainAccount extends FormRun {
    ...
    public const int listCreateRoot = 1;
    ...
}
```

**Step 2. Build the context menu**

Construct the list of submenus and menu options, and add it to the control's context menu.

1. Add the `getContextMenuOptions()` method override on the control.

2. Create a new context menu and a list to hold the options that you will add to the menu:
Step 3. Process the user selection from the context menu

1. Add the `selectedMenuOption()` method override on the control.
2. Move the `switch()` statement for processing options into this override.

**Code example**

This section illustrates the migration of a context menu from Dynamics AX 2012 to Finance and Operations. The Main Account form is used as an example.

**Original code**

```java
public void context()
{
    PopupMenu listMenu = new PopupMenu(element.hWnd());
    int listCreateRoot = listMenu.insertItem("@SYS5480");
    int selectedMenu;
    selectedMenu = listMenu.draw();
    switch (selectedMenu)
    {
    case -1:
        break;
    case listCreateRoot:
        mainAccount_ds.create();
        break;
    default:
        break;
    }
}
```

**Migrated code**

```java
ContextMenu menu = new ContextMenu();
List menuOptions = new List(Types::Class);

3. Add menu options to the list:

   - ContextMenuOption option = ContextMenuOption::Create(label, identifier);
   - menuOptions.addEnd(option);

4. Add the list of options to the menu.

   - menu.ContextMenuOptions(menuOptions);

5. Modify the return statement.

   - return menu.Serialize();
```
// Define new form-level constant for each context menu option
public const int listCreateRoot = 1;
// Define new override on the control for building the context menu
public str getContextMenuOptions()
{
    str ret;
    ContextMenu menu = new ContextMenu();
    ContextMenuOption option = ContextMenuOption::Create("@SYS5480", listCreateRoot);
    List menuOptions = new List(Types::Class);
    // Add label and ID of menu option
    menuOptions.addEnd(option);
    menuContextMenuOptions(menuOptions);
    return menu.Serialize();
}
// Define new override on the control for processing the user selection
public void selectedMenuOption(int selectedOption)
{
    switch (selectedOption)
    {
    case -1:
        break;
    case listCreateRoot:
        mainAccount_ds.create();
        break;
    default:
        break;
    }
}

UX guidelines for context menus

As you migrate context menus, consider the following guidelines:

- The most important commands should be at the top of the menu.
- Remove commands that don't apply to the current state of the element that is the target of the right-click.
- Right-click is a shortcut. Therefore, the commands on the context menu should always be available in other places on the page.
- Don't create submenus of context menus. Submenus are hard to use and aren't touch-friendly.
- Limit the number of menu items to five.
In Finance and Operations, the `mouseDblClick()` override has been deprecated, and you will need to move this logic to new controls.

In Microsoft Dynamics AX 2012, the mouse double-click event was used for various reasons. For example, it helped provide a better user experience and provided an alternative way to run certain scenarios. Here are some examples of common usage patterns:

- Moving elements between two lists or tree controls
- Opening a new form to get more details about the selected field
- Running complex business logic
- Selecting a field in a lookup

**Strategy overview**

Before you begin to use the form, it's a good idea to fix all best practice warning messages that state, “The mouseDblClick control method has been deprecated and should not be used.” Otherwise, the form might be useless, or it might work only in limited ways.

**Migrate code from mouseDblClick() methods**

As we mentioned earlier, there were various reasons for using the `mouseDblClick()` method in Dynamics AX 2012. This section explains how to migrate some of the most common scenarios.

**Moving items between two lists controls**

In Dynamics AX 2012, a mouse double-click was often used in List Panel scenarios, where two list controls appeared side by side. Often, when a user double-clicked an item in one list control, that item was moved to the second list control. Migration of this `mouseDblClick()` scenario involves alignment to the List Panel pattern. You have two options for migrating this usage pattern:

- Use the `SysListPanel` class itself, which provides the logic and the buttons for moving items between the two list controls.
- If you can't use the `SysListPanel` class (because the lists aren't ListViews, or the class isn't appropriate for the given situation), you can manually model the controls by following the List Panel sub-pattern. This pattern includes buttons for moving items between lists, but the developer will have to add the correct logic to make these buttons work.

**Opening a new form**

In another common usage pattern in Dynamics AX 2012, the user double-clicked a field to open a new form that showed more detailed information about that field. You have several options for migrating this usage pattern:

- Use a single-click to open a backing form that shows more details about a field. This functionality is automatically implemented for many fields that are based on table relations, and you can implement it manually by overriding the `jumpRef()` method on a control. The preferred migration route is to move the code from `mouseDblClick()` into a `jumpRef()` override, so that the navigation will be aligned with other fields in the system.
- Model a new button on the form, and move the logic from the `mouseDblClick()` method into the button's `clicked()` method. You should use this approach only for non-input field controls (for example, a Tree
control) in which a `jumpRef()` override doesn't exist.

- Add a right-click context menu (shortcut menu) option. However, note that UX guidelines specify that the commands on context menus should **always** be available in other locations on the page. A `View details` command is automatically added to the right-click context menu for controls that have an overridden `jumpRef()` method. Therefore, this approach should be used only as an optional addition to the previous migration route (modeling a new button). For more information about how to add context menu options, see Code migration - Context menu code.

**Moving logic to a button control**

In another common usage pattern in Dynamics AX 2012, the double-click caused complex business logic to run. For this scenario, the preferred migration route is to model a new button on the form, and then move the logic from `mouseDblClick()` into the new button's `clicked()` method.

**Selecting a field in a lookup**

In some custom lookups in Dynamics AX 2012, code was added so that the user could double-click a row in a grid (or an element in a tree) to select the value and close the lookup. For this scenario, the recommended migration route is to add a **Select** button at the bottom of the lookup form to enable record selection.

**UX guidelines**

As you migrate mouse double-click methods, you should consider the following guidelines:

- To move items between controls, use the `SysListPanel` class or the ListPanel pattern whenever possible.
- When you add buttons to replace mouse double-click logic, put the button as close as possible (contextually) to the control.
- In some cases, you might have to redesign the form to accommodate the logic that was present in the `mouseDblClick()` method.
In data entry scenarios, it is common for a user to attempt to identify an entity in terms of some more descriptive or natural language attribute if that entity is formally identified by a synthetic key, such as a number sequence. The contextual data entry feature allows users to type in either the synthetic key or a more descriptive attribute directly into a lookup field. This page explains how contextual data entry works and also provides implementation details and tips for developers who want their lookups to have this behavior.

Introduction

In data entry scenarios, it is common for a user to attempt to identify an entity in terms of some more descriptive or natural language attribute if that entity is formally identified by a synthetic key, such as a number sequence. A user will typically attempt to enter an Account Name instead of an Account ID for the Customer Account when creating a Sales Order. This is because most interaction with a customer is done using their actual name instead of some synthetic identifier. Unfortunately, any user’s attempt to enter an Account Name will fail because the Customer account control’s underlying foreign key relates to a field that is a synthetic key—a number sequence—and Dynamics AX 2012 (and older) will always attempt to validate the entered value directly. Therefore, if the Account ID was unknown to the user, the user would be forced to perform some type of searching step, such as opening the Customer account control’s lookup and filtering on the Account Name column to identify the correct Account ID (see the image below).

Create sales order

![Customer account lookup](image)

This user experience is not optimal and is being addressed by data entry efficiency and productivity. The platform adds initial support for contextual data entry, where the system automatically attempts to understand whether the user’s entered data is in the context of the key field or some other more descriptive or well-understood field, and handle it appropriately. For the remainder of this document, we’ll generically refer to these types of fields as ID (synthetic) and NAME (descriptive) fields, respectively.

Contextual lookup forms

Just like keyboard data entry, all system-generated lookup forms are also now contextual, meaning that filtering and sorting occur in the context of the data the user has entered. Using the create a Sales Order scenario as an example, the user will see the lookup shown below if an ID is entered.
If a NAME is entered, then the user will see the following lookup. Notice how the NAME column is moved first in the Grid, and how the lookup is sorted and filtered upon when the user’s data is in the context of NAME.

Contextual data entry implementation details

**Behavior**

In the context of the Sales order create scenario mentioned above, the contextual data entry feature will allow the user to be able to freely type in either the ID or NAME without performing any laborious search process. In detail, the following behaviors will occur:

1. If the user enters a complete ID reference, the value will be taken directly.
2. If the user enters a complete and unique NAME reference, the value will be automatically translated into an ID and then processed.
3. If the user enters a non-complete ID or NAME reference (such as Micro instead of Microsoft), but it still uniquely matches either ID or NAME via a BEGINS WITH predicate, then the value will be translated into its complete ID and then processed.
4. If the user enters a non-complete ID or a non-unique NAME and there are multiple matches, then a disambiguation lookup will be presented to the user to select which value was actually intended.

See Appendix A for more detailed sample scenarios of contextual data entry.

**Prerequisites**

To maintain functional correctness and reasonable performance, the following constraints were added to the application of the behaviors described in the previous section:

1. **Title Field 2** is the NAME field**.
2. The NAME field must either be covered by an index OR belong to a Table whose **Cache Lookup** property is set to **EntireTable**. All contextual lookup behavior will be disabled if this requirement is not met for performance reasons. **NOTE: An index should only be added for NON TRANSACTIONAL tables**
because of index maintenance costs. Also note that you will likely want to mark this index as non-unique (Allow Duplicates = Yes).

3. If a control is using a custom lookup form (such as SysTableLookup; FormHelp on an EDT) then the disambiguation behavior described previously will not be turned on by default. This is because these custom lookup forms (and even surrounding modified and lookup method overrides) can and will do advanced things such as presenting a dialog, which are not desirable in the context of contextual lookups.

Handling custom lookup forms requires additional knowledge and will be covered in its own section.

**Programming model additions**

The behaviors and rules expressed in Listings 1 and 2 are contained primarily by a new X++ class called FormControlAmbiguousReferenceResolver. FormControlAmbiguousReferenceResolver uptake in application code will be necessary in more advanced scenarios. Its use will be described later in the document. In addition to the FormControlAmbiguousReferenceResolver class, a new control override called `resolveAmbiguousReference` has been added. `resolveAmbiguousReference` acts as a hook point in the system for translating what the user typed into a value that the system is expecting. The basic flow is as follows:

1. The user enters a value into a control and removes focus.
2. An interaction is sent from the client to the server, indicating that a new value has been entered. The appropriate command is executed on the server.
3. Before the command attempts to process the value entered by the user, it makes a call to `resolveAmbiguousReference` to give the system a chance to translate the value into the expected domain.
4. The super implementation of `resolveAmbiguousReference` creates an instance of FormControlAmbiguousReferenceResolver which executes the rules described above.

The value returned from `resolveAmbiguousReference` is used for the remainder of the command’s execution. `Validate()` and `modified()` operate against the returned value.

**Standard lookup uptake**

**Add an index that covers TitleField2**

`TitleField2` defines the default definition of NAME. In order to enable ID and NAME contextual data entry, `TitleField2` must be either indexed OR belong to a table with `CacheLookup` set to `EntireTable`. If the table containing `TitleField2` does not yet define an index covering `TitleField2` and, importantly, the table does not have a high volume of CUD (Creates/Updates/Deletes*), then add a non-unique index (Allow Duplicates = Yes) covering `TitleField2`. This will cause the system to start executing the contextual data entry behavior, except for the custom lookup limitation described in the Prerequisites section. *Adding an index on high-volume transactional tables may incur a noticeable performance penalty due to index maintenance costs.

**Enable disambiguation behavior for custom lookup scenarios**

Custom lookup implementations can provide advanced or non-typical behaviors, such as presenting dialogs. Therefore, the system disables the default disambiguation behavior when a custom lookup scenario is detected. To opt into the default disambiguation behavior, override the `resolveAmbiguousReference` method (as shown below) on the control hosting the lookup. Note that the second parameter to the `resolveAmbiguousReferenceForControl` call is what overrides the default behavior of not performing disambiguation for custom lookup scenarios.

```csharp
public str resolveAmbiguousReference()
{
    FormControlAmbiguousReferenceResolver::resolveAmbiguousReferenceForControl (this, true);
}
```

**Make custom lookup forms contextual**
As mentioned earlier, all system-generated lookup forms automatically consider the context of the data entered into their host control. This includes most lookup forms generated via `SysTableLookup`. Modeled custom lookup forms, by their nature, cannot be fully-handled by the system and must be modified to match the behavior and visuals of contextual lookups forms.

1. If the data contained by the host control is in the context of ID, then:
   a. Make the ID column first in the Grid.
   b. Sort and filter by ID.
2. If the data contained by the host control is in the context of NAME, then:
   a. Make the NAME column first in the Grid,
   b. Sort and filter by NAME,

The following scenarios illustrate some custom lookups, along with the recommendation for how to enable contextual data entry in these cases.

**Scenario 1: Custom lookup defined via the FormHelp property on an EDT**

Custom lookups defined via FormHelp (even though modeled) still go through normal kernel-based lookup generation routines. Therefore, the kernel still has hooks to make some changes to the lookup form. Specifically, the lookup system has enough information to apply the correct filters and sorts; however, it is NOT known which controls should be moved in the lookup's grid. (While an educated guess could be made based on bindings, that guess may be incorrect in more advanced lookup form designs.) If your custom lookup form is leveraging the `SysTableLookup::filterLookupPreRun` and `SysTableLookup::filterLookupPostRun` methods, then uptake the (new) optional parameters on `filterLookupPostRun` to have the NAME control moved automatically, as shown.

```java
public class MyCustomLookupForm extends FormRun
{
  public void run()
  {
    FormStringControl lookupHostControl = SysTableLookup::getCallerStringControl(this.args());
    boolean isFiltered = SysTableLookup::filterLookupPreRun(lookupHostControl, ID_Control,
    FormDataSourceToFilter);
    super();
    SysTableLookup::filterLookupPostRun(isFiltered, lookupHostControl.text(), ID_Control,
    FormDataSourceToFilter,
    new FormControlAmbiguousReferenceResolver(callingControl), NAME_Control);
  }
}
```

If your lookup form isn’t using the `SysTableLookup::filterLookup*` methods, and you don’t want to uptake those methods, then you can simply add a control move as shown below.
public class MyCustomLookupForm extends FormRun
{
    public void init()
    {
        super();
        this.applyControlOrdering();
    }

    private void applyControlOrdering()
    {
        FormControl callerControl = SysTableLookup::getCallerControl(this.args());
        if (FormControlAmbiguousReferenceResolver::isControlValueMappedToAlternativeField(callerControl))
        {
            Grid.moveControl(ID_Control.id(), NAME_control.id());
        }
        else
        {
            Grid.moveControl(NAME_Control.id(), ID_Control.id());
        }
    }
}

Scenario 2: Override of lookup method manually launching a form

Unlike Scenario 1, lookup forms launched by completely manual mechanisms, such as the class factory, have no kernel hooks. Therefore, it is the responsibility of the lookup form to adhere to the contextual data entry behaviors. The easiest way to do this is to leverage the SysTableLookup::filterLookup* methods (similar to Scenario 1) except include one additional parameter to indicate that sorting should also be maintained. An example is shown below.

public class MyCustomLookupForm extends FormRun
{
    public void run()
    {
        FormStringControl lookupHostControl = SysTableLookup::getCallerStringControl(this.args());
        boolean isFiltered = SysTableLookup::filterLookupPreRun(lookupHostControl, ID_Control,
        FormDataSourceToFilter);
        super();
        SysTableLookup::filterLookupPostRun(isFiltered, lookupHostControl.text(), ID_Control,
        FormDataSourceToFilter,
        new FormControlAmbiguousReferenceResolver(callingControl), NAME_Control, true);
    }
}

Advanced lookup uptake

Scenario 1: Overriding ID and NAME bindings

If you want to use a set of fields other than what is chosen by default, you must manually construct an instance of FormControlAmbiguousReferenceResolver and provide the optional parameters representing the custom bindings. This specialized instance must be used in an override of resolveAmbiguousReference and in a custom lookup form (including SysTableLookup, which also accepts an instance of FormControlAmbiguousReferenceResolver). A custom binding cannot currently be specified in kernel-generated lookups. Methods currently accepting custom ID and NAME bindings:

1. FormControlAmbiguousReferenceResolver
   - Constructor
   - resolveAmbiguousReferenceForControl
Here's an end-to-end example of how to provide custom bindings.

```java
public str resolveAmbiguousReference()
{
    return FormControlAmbiguousReferenceResolver::resolveAmbiguousReferenceForControl(
        this, true, AbsoluteFieldBinding::construct(IDField, Table),
        AbsoluteFieldBinding::construct(SomeOtherNAMEField, Table));
}
```

```java
public class MyCustomLookupForm extends FormRun
{
    public void run()
    {
        FormStringControl lookupHostControl = SysTableLookup::getCallerStringControl(this.args());
        boolean isFiltered = SysTableLookup::filterLookupPreRun(lookupHostControl, ID_Control,
        FormDataSourceToFilter);
        super();
        SysTableLookup::filterLookupPostRun(isFiltered, lookupHostControl.text(), ID_Control,
        FormDataSourceToFilter,
        new FormControlAmbiguousReferenceResolver(callingControl,
            AbsoluteFieldBinding::construct(IDField, Table),
            AbsoluteFieldBinding::construct(SomeOtherNAMEField, Table)), NAME_Control, true);
    }
}
```

**Scenario 2: Custom resolution logic**

It's possible to use custom resolution logic by overriding resolveAmbiguousReference and leveraging something other than FormControlAmbiguousReferenceResolver. Note that this logic needs to be common to the hosted lookup form so that keyboard and lookup-based entry stay in sync.

```java
public str resolveAmbiguousReference()
{
    // In this sample, allow “looser” data entry by simply picking the first record that matches, if any.
    CLI_Job _job;
    str mappedValue = this.text();
    if (strLen(mappedValue) > 0)
    {
        select firstonly _job order by _job.Title where _job.Title like mappedValue + "*";
    }
    if (_job.RecId)
    {
        mappedValue = _job.Title;
    }
    return mappedValue;
}
```

**Appendix Detailed usage scenarios for contextual data entry**

For the scenarios, assume there is a table called "TableA" with PK field "ID" and index field "Name", with the FK we're trying to enter that is related to the ID (the user ultimately needs to pick an ID). Note that any algorithms that depend on like/begins with are assuming string fields. We won't be able to provide high fidelity resolution behavior on, for example, integral types.
Scenario 1: User enters a valid ID of "1234" The super() implementation of resolveReference first queries against TableA.ID with the appropriate predicate. The query finds a single record, and returns the user's entered value to be further processed by validate and modified. Validation passes and the user sees "1234" in the UI.

Scenario 2: User enters an invalid ID of "4321" The super() implementation of resolveReference first queries against TableA.ID. The query does not find any records, so a second query is performed against the Name field (SELECT TOP 2 FROM TableA WHERE TableA.Name LIKE "4321%"). Still, no record is found, so "4321" is passed through to validation, which fails. The user sees "4321" in the browser as well as a validation error.

Scenario 3: User enters a valid Name of "ACME" The super() implementation of resolveReference first queries against TableA.ID. The query does not find any records, so a second query is performed against the Name field (SELECT TOP 2 FROM TableA WHERE TableA.Name LIKE "ACME%"). This query does find a single record (unique reference), so the lookup automatically returns the corresponding TableA.ID. Validate and modified continue executing in the context of that value. Validation passes, and ultimately the user sees the ID value for ACME in the browser (for example, ACME would switch to 1234 in the browser).

Scenario 4: User enters an invalid Name of "ACNE" The super() implementation of resolveReference first queries against TableA.ID. The query does not find any records, so a second query is performed against the Name field (SELECT TOP 2 FROM TableA WHERE TableA.Name LIKE "ACNE%"). This query does not find any records, so the lookup passes ACNE through to validation, which fails. The user sees "ACNE" in the browser as well as a validation error.

Scenario 5: User enters an ambiguous Name of "ACME" In this case, assume there are two records in the database: one with Name "ACME W" and another with "ACME E". The super() implementation of resolveReference first queries against TableA.ID. The query does not find any records, so a second query is performed against the Name field (SELECT TOP 2 FROM TableA WHERE TableA.Name LIKE "ACME%"). This query finds two records, so it cannot make any further assumptions. A disambiguation lookup is presented to the user showing "ACME W" and "ACME E" as choices. The user picks "ACME E". resolveReference then takes the records selected by the user and redirects it to the ID of "ACME E". Validate and modified continue execution in the context of the ID of "ACME E". The browser ultimately displays the ID of "ACME E" (for example, 1234).

Scenario 6: User enters an ambiguous Name of "ACME" and doesn't make a choice in the disambiguation lookup In this case, assume there are two records in the database: one with Name "ACME W" and another with "ACME E". The super() implementation of resolveReference first queries against TableA.ID. The query does not find any records, so a second query is performed against the Name field (SELECT TOP 2 FROM TableA WHERE TableA.Name LIKE "ACME%"). This query finds two records, so it cannot make any further assumptions. A disambiguation lookup is presented to the user showing "ACME W" and "ACME E" as choices. The user doesn't make a selection from the lookup. Therefore "ACME" is passed through to validate and modified. Validation fails and the user is presented with a validation failure message. The browser still displays a value of "ACME".

Scenario 7: User enters a "valid" ID of "12" and presents the lookup form Prior to presenting the lookup, the system queries against TableA.ID (SELECT TOP 1 FROM TableA WHERE TableA.ID LIKE '12%'). The query finds a record and therefore assumes the user must be operating in the context of ID. It presents the lookup, filtering and sorting by ID.

Scenario 8: User enters an invalid ID of "4321" and presents the lookup form Prior to presenting the lookup, the system queries against TableA.ID (SELECT TOP 1 FROM TableA WHERE TableA.ID LIKE '4321%'). The query does not find a matching record and therefore assumes the user is entering a Name. The lookup is presented as filtered and sorted by Name (no records shown in this case).

Scenario 9: User enters a "valid" Name of "AC" and presents the lookup form Prior to presenting the lookup, the system queries against TableA.ID (SELECT TOP 1 FROM TableA WHERE TableA.ID LIKE 'AC%'). The query does not find a matching record and therefore assumes the user is entering a Name. The lookup is presented as filtered (those records matching "begins with AC") and sorted by Name in alphabetical order.
Scenario 10: User enters an invalid Name of "EM" and presents the lookup form
Prior to presenting the lookup, the system queries against TableA.ID (SELECT TOP 1 FROM TableA WHERE TableA.ID LIKE 'EM%'). The query does not find a matching record and therefore assumes the user is entering a Name. The lookup is presented as filtered and sorted by Name. No records are found and therefore the user is presented with an empty lookup.
This article provides information about the HierarchyViewer control, which lets you represent hierarchical relationships for people, products, or organizations.

**Overview**

The HierarchyViewer control lets you represent hierarchical relationships for people, products, or organizations. It's used primarily as a graphical means to help you understand hierarchical relationships in a traditional top-down manner, and as a way to navigate to the entity that is represented by the focused node. The HierarchyViewer control lets you walk through deeply nested, multilevel content in a compact space. The control expands and collapses nodes to control the parts of the tree structure that are shown. Because it's an unbound control, the HierarchyViewer data is managed by an abstraction class and is used primarily as a way to visualize data in a simple tree relationship. For hierarchy data in a traditional tree, there is a standard tree control.

The HierarchyViewer control shows four levels of information at any given time. The current node is the current focus of the tree, which is not necessarily the root node. The current node is represented by the largest physical node in the current view and it has a colored bar on the left. Above the current node is a trail of smaller parent nodes from the root node down to the current node. Below the current node is a level of children nodes, and there can be an indefinite number of nodes at this level. By default, three children nodes are shown at a time on each page, but that can changed by adjusting the Number of children property. The Next and Previous link buttons allow the user to page to other nodes at the child level. Finally, there is a level of grandchildren nodes that are shown for each child node. Each child can have an indefinite number of grandchild nodes, and the number of grandchildren shown at one time for each child node is controlled by the Number of

---

**NOTE**

This visual is available starting with platform updates for version 10.0.22 of Finance and Operations.
Grandchildren property. Users can use the Next and Previous arrow buttons to page up and down through members at the grandchild level. The interactive display of nodes requires no business logic.

**Business logic interaction**

The HierarchyViewer control offers data visualization and navigation. The HierarchyViewer control is a read-only control. It can be used to select an entity (employee, product, or organization), and corresponding data can then be managed though other display and input fields on the form, outside the HierarchyViewer control. This is accomplished by a selection event that is raised on each user focus on each node.

```java
public void init()
{
    // HierarchyViewer is the auto-declared name for the control.
    // handleNodeSelected is your event handler.
    HierarchyViewer.notifyNodeSelected += eventhandler(element.handleNodeSelected);
}
public void handleNodeSelected(int _nodeId)
{
    // do something
}
```

**Authoring a HierarchyViewer instance**

To create a HierarchyViewer instance:

1. In the form designer, add an instance of HierarchyViewer to your form.
2. In the Properties pane, accept the default number of visible children and grandchildren, or set new values.

The HierarchyViewer control is primarily a visually interactive way of navigating or interrogating nodes in a static manner. The HierarchyViewer control isn’t bound to a data source. Instead, the control is managed by a corresponding controller class that extends the base `HierarcyDesignerBase`. You initialize that class with data, and bind to the control instance and the visible fields of the HierarchyViewer node.

A typical use of the control is to initialize a server-side “in-memory” map of the hierarchy and then dynamically update the control as the user interactively explores the hierarchy by using load-on-demand semantics.
public void init()
{
    HcmPositionNode node;
    nodeMap = new Map(Types::Int64, Types::Class);
    hierarchyMap = new Map(Types::Int64, Types::Int64);
    firstNodeId = 0;
    // Initialize the organization node
    node = HcmPositionNode::newParameters(this.getNextNodeId(), HcmPositionNodeType::Enterprise, -1, 0,
                                           "@SYS317690", """);
    rootNode = node;
    if (selectedNode == null)
    {
        selectedNode = rootNode;
    }
    this.insertNewNodeAndUpdateParent(node);
}

public void applyBuild()
{
    super();
    YourControllerClass controller = new YourControllerClass();
    this.initControl(controller);
}

public void initHcmPositionFromCurrentNode(HcmPosition _hcmPosition)
protected void insertNewNodeAndLoadDescendants(HcmPositionNode _node, int _depth, HcmPositionNode _parentNode = null, Common _common = null)
protected void loadNodeDescendants(HcmPositionNode _node, int _depth, Common _common = null)

Changing node visuals

You can't change node visuals. The design presents a consistent visual and user interaction.
This article discusses how to enable lookup behavior on controls. It also discusses how to create multi-select lookups and outlines lookup scenarios that are no longer supported.

**Enabling lookup behavior in controls**

**Controls bound to an Extended Data Type**

Controls with their Extended Data Type property set (no FormDataSource in play) will have a lookup under the following conditions:

1. If the EDT has its Table Relations or Table References node populated.
2. If the FormHelp property is set (custom lookup); doesn’t require rule #1 to be true.
3. If the control has lookup or lookupReference overridden. Note, this rule also applies to fully unbound controls (no EDT, field, or data method). This includes overrides via registerOverrideMethod and others.

**Controls bound to a form data source**

Controls that are bound to a data source will have a lookup under the following conditions: **Field bound**

1. “lookup” or “lookupReference” (Reference Controls) methods are overridden.
   a. If the FormDataSource field has lookup or lookupReference overridden.
   b. If the control has lookup or lookupReference overridden.
      ● This includes overrides via registerOverrideMethod and others.
2. If the field has an EDT, then rule #2 from the “Controls bound to an Extended Data Type” section applies.
3. If the bound field maps to a relation per DBFGetRef rules.
   a. High level rules:
      a. If there is an EDT relation backing the field, with the Table Relations node populated and Ignore EDT Relations is false on the field, the relation is used (has a lookup).
      b. If there is a relation mapping to the field and any fixed field link conditions are satisfied, the relation is used (has a lookup).
         a. Validate must be “Yes”.
         c. Note the special case of migrated EDT relations which occur when:
            a. Field is backed by an EDT with the Relations node populated.
            b. Field is backed by a TABLE relation with the “EDTRelation” set to Yes.
            c. The table relation link has the SourceEDT set to the appropriate EDT.
            d. You can also have cases where IgnoreEDTRelation is set to true on a field, in which case a lookup will occur only if rule #3.1.2 of this section is true.

**Data method bound**

1. If the return type of the data method is an EDT, then rules #1 and #2 from the “Controls bound to an Extended Data Type” section apply.
2. If the control has lookup or lookupReference overridden.
   ● This includes overrides by using registerOverrideMethod.
Multiselect lookups

Available system forms for building multi-select lookups

There are currently two system forms for creating multi-select lookups:

- SysLookup – Multiselect based on an Enum.
- SysLookupMultiselectGrid – Multiselect based on a collection of data.

What happened to the SysLookupMultiselect form?

SysLookupMultiselect was marked for deprecation in Microsoft Dynamics AX 2012 and has been removed. Any use of this form for multiselect lookup scenarios should be migrated to use SysLookupMultiselectGrid. For an example, see the form tutorial_LookupMultiSelectGrid.

Unsupported lookup scenarios

Creating multiple lookup forms when the lookup button is used

An error may occur if you create multiple lookup forms when the lookup button is used. For example, overriding the 'lookup' method and creating a new lookup form, but also calling 'super' (which will create another lookup form).

Using SelectedControl() to determine which control is hosting a lookup

Using SelectedControl() to determine which control is hosting a lookup is unsupported. While it may work in some cases, it will fail in others. For example, in disambiguation lookups, no control is selected on the parent form since the act of leaving the control is what triggers a disambiguation lookup. As an alternative to using SelectedControl(), there are a few other ways to retrieve the control that is hosting the lookup:

- Check the 'selectTarget' of the lookup form.

  ```java
  FormStringControl selectTarget = formRun.selectTarget();
  ```

- Check the 'callerFormControl' on the lookup form args. Note that SysTableLookup::getCallerControl(Args args) encapsulates that call.

  ```java
  FormStringControl argsCallerFormControl = args.callerFormControl();
  ```

Note that the selectTarget and callerFormControl will be set automatically if the lookup form instance is spun up automatically by the kernel. If the form instance is created in app code, these can be set manually as shown below.

```java
public void lookup()
{
    Args args = new Args(formStr(<formName>));
    args.caller(element);
    args.callerFormControl(this);
    FormRun formRun = classfactory.formRunClass(args);
    formRun.init();
    this.performFormLookup(formRun);
}
```

Creating a slider dialog (instead of a lookup form) when the lookup button is used

Lookup controls should open lookup forms when the lookup button is used (not slider dialogs or other kinds of forms). The first reason for this is product consistency. The second and more important reason is that opening a slider dialog from a lookup is incompatible with the new type-ahead feature in lookups.
This topic provides information about the file upload control. This control lets users upload files.

Overview

The file upload control lets users upload a file. It also lets developers control the upload process and manage the file that is uploaded, based on their requirements.

The file upload control can have three styles. You control the style by using the **Style** property.

- The **Standard** style shows the file name field together with **Browse**, **Upload**, and **Cancel** buttons.
- The **Minimal** style shows only the **Browse** button.
- The **MinimalWithFileName** style shows the file name field and the **Browse** button.

The **FileTypesAccepted** property of the file upload control lets you limit the types of files that users can upload. The file types that users can upload are primarily controlled by the associated upload strategy. The **FileTypesAccepted** property on the file upload control should be used only if further restrictions are required. If the upload control tries to specify file types that are restricted by the upload strategy, the **Browse** button becomes unavailable.

<table>
<thead>
<tr>
<th>ALLOWED FILE TYPES</th>
<th>ALLOWED FILE TYPES FROM THE UPLOAD STRATEGY</th>
<th>FINAL RESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;jpg.png&quot;</td>
<td>&quot;jpg.png,.gif,.txt&quot;</td>
<td>&quot;jpg.png&quot;</td>
</tr>
<tr>
<td>&quot;image/png&quot;</td>
<td>&quot;image/*&quot;</td>
<td>&quot;image/png&quot;</td>
</tr>
<tr>
<td>&quot;image/*&quot;</td>
<td>&quot;image/png&quot;</td>
<td>The <strong>Browse</strong> button is unavailable.</td>
</tr>
<tr>
<td>&quot;jpg.png,.gif,.txt&quot;</td>
<td>&quot;jpg.png&quot;</td>
<td>The <strong>Browse</strong> button is unavailable.</td>
</tr>
</tbody>
</table>

You can use the **OnBrowseButtonClicked**, **OnUploadAttemptStarted**, and **OnUploadCompleted** overrides to hook into the various stages of the file upload process. You can also create custom file upload strategies and associate them with a file upload control by using the **FileUpload Strategy Class** property.

Design classes

There are two base classes that developers can work with for the file upload control:

- **Upload strategy class** – This base class lets developers control various parameters that should be enforced for uploaded files, such as the types of files that a user can upload and the maximum size of a file. It also lets developers determine where and how the uploaded file should be stored. All derived classes used for upload strategies must inherit from the abstract **FileUploadStrategyBase** class.
- **Upload result class** – This base class lets developers access the details of a file that was uploaded by a user,
such as its name, content type, and upload status. It also lets developers open and delete the corresponding file. All derived classes used for specializing upload results must inherit from the abstract **FileUploadResultBase** class.

The framework provides a default upload strategy class that is named **FileUploadTemporaryStorageStrategy** and a default upload result class that is named **FileUploadTemporaryStorageResult**. This upload result class stores uploaded files to the temporary blob storage and provides a download URL. Developers can also implement their own custom upload strategy and upload result classes as required. For the upload strategy, two abstract methods from the **FileUploadStrategyBase** class must be implemented: **uploadFile** and **getResultClassName**. The **uploadFile** method handles where and how the file is stored. The **getResultClassName** method retrieves the upload result class that is used in this strategy. The **FileUploadResultBase** class has fields for the file name, the upload status, the content type of the file, and the log message. This class can be extended as required. All new properties should be able to be serialized and deserialized. The **openResult** method opens the file as a stream, and the **deleteResult** method deletes the file from the corresponding data storage.

**Sequence diagram**

The file upload control accepts the file and upload strategy in the client, and sends them to the file services. The file services start a new session, create an instance of a strategy class, and call the **uploadFile** method. When the **uploadFile** method has finished storing the file in the data source, a file upload result class returns to the file services. This class is sent back to the client, which might trigger the **OnUploadCompleted** event to deal with the post-process.

**Scanning uploaded files for viruses and malicious code**

Before you upload a file into the system, you might want to scan it for viruses or malicious code. Therefore, in version 10.0.12 and later, an extension point is available so that customers can integrate the file scanning software of their choice into the file upload process. Similar extension points are also available for scanning attachments. For more information about those extension points, see [Configure document management](#).
**IMPORTANT**

Out of the box, Finance and Operations apps don't scan files for viruses and malicious code, and we don't recommend specific software for file scanning. Instead, customers are responsible for choosing their own file scanning software, and for adding the appropriate code to the delegate handlers so that they can use the software or service of their choice to scan files.

In particular, the `FileUploadResultBase` class exposes the `delegateScanStream()` delegate. This delegate applies to any file upload scenario where the **Upload strategy class** has been specialized. The upload process will fail if the scanning service determines that the file is malicious.

**Implementation details**

The following example of the `ScanDocuments` class shows boilerplate code for the handler. For general information about how to implement handlers for delegates, see **EventHandlerResult classes in request or response scenarios**.

```csharp
public final class ScanDocuments
{
    [SubscribesTo(classStr(FileUploadResultBase, staticDelegateStr(FileUploadResultBase, delegateScanStream))]
    public static void FileUploadResultBase_delegateScanStream(System.IO.Stream _stream, EventHandlerRejectResult _validationResult)
    {
        if (!ScanDocuments::scanStream(_stream))
        {
            _validationResult.reject();
        }
    }

    private static boolean scanStream(System.IO.Stream _stream)
    {
        /*
        Custom implementation required for connecting to a scanning service
        If document scanning process found an issue, return false; otherwise, return true;
        */
        return true;
    }
}
```

Out of the box, Finance and Operations apps don't scan files for viruses and malicious code, and we don't recommend specific software for file scanning. Instead, customers are responsible for choosing their own file scanning software, and for adding the appropriate code to the delegate handlers so that they can use the software or service of their choice to scan files.

In particular, the `FileUploadResultBase` class exposes the `delegateScanStream()` delegate. This delegate applies to any file upload scenario where the **Upload strategy class** has been specialized. The upload process will fail if the scanning service determines that the file is malicious.

**Implementation details**

The following example of the `ScanDocuments` class shows boilerplate code for the handler. For general information about how to implement handlers for delegates, see **EventHandlerResult classes in request or response scenarios**.

```csharp
public final class ScanDocuments
{
    [SubscribesTo(classStr(FileUploadResultBase, staticDelegateStr(FileUploadResultBase, delegateScanStream))]
    public static void FileUploadResultBase_delegateScanStream(System.IO.Stream _stream, EventHandlerRejectResult _validationResult)
    {
        if (!ScanDocuments::scanStream(_stream))
        {
            _validationResult.reject();
        }
    }

    private static boolean scanStream(System.IO.Stream _stream)
    {
        /*
        Custom implementation required for connecting to a scanning service
        If document scanning process found an issue, return false; otherwise, return true;
        */
        return true;
    }
}
```
This topic describes the system-defined buttons.

Overview

Several system-defined buttons are automatically present on the Action Pane. In general, these system-defined buttons should be applicable and should kept available to the end user. However, in rare cases (for example, if a more specialized control is required, or if a system-defined button isn’t useful or applicable for a particular form), developers might have to explicitly suppress or override a system-defined button. For example, in some situations, a MenuButton that lets the user select from multiple “New” options might be preferable to the system-defined New button.

List of system-defined buttons

The following tables give the full list of system-defined buttons. The tables also provide information that will be useful if these buttons must be conditionally or completely suppressed or overridden.

Common buttons

<table>
<thead>
<tr>
<th>BUTTON</th>
<th>BUTTON NAME MACRO*</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export</td>
<td></td>
<td>Don't suppress this button.</td>
</tr>
<tr>
<td>Attach</td>
<td>#SystemDefinedAttachButton</td>
<td>Don't suppress this button, because we will suppress it on forms that aren't set up for attachments.</td>
</tr>
<tr>
<td>Show filters</td>
<td>#SystemDefinedShowFiltersButton</td>
<td>By default, Visible=No on TOC forms.</td>
</tr>
</tbody>
</table>

* System-defined button name macros are found in the SysSystemDefinedButtons macro file.

Buttons that are specific to Details forms

These buttons are an integral part of the Details form experience. Therefore, it's very unlikely that you'll have to suppress these buttons.

<table>
<thead>
<tr>
<th>BUTTON</th>
<th>BUTTON NAME MACRO*</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change view</td>
<td>#SystemDefinedShowMenuButton</td>
<td>This button exists under the Options Action Pane tab.</td>
</tr>
<tr>
<td>Grid view</td>
<td>#SystemDefinedGridViewButton</td>
<td></td>
</tr>
<tr>
<td>Details view</td>
<td>#SystemDefinedDetailsViewButton</td>
<td></td>
</tr>
<tr>
<td>Line details view</td>
<td>#SystemDefinedLineDetailsViewButton</td>
<td></td>
</tr>
<tr>
<td>Header details view</td>
<td>#SystemDefinedHeaderDetailsViewButton</td>
<td></td>
</tr>
</tbody>
</table>
Form styles that have no system-defined buttons

For several form styles, it doesn’t make sense to add system-defined buttons, primarily because these forms don’t have standard Action Panes. The following form styles never receive system-defined buttons:

- Dashboard
- Dialog
- DropDialog
- FormPart
- Lookup
- Sitemap
- Wizard

Suppressing most or all of the system-defined buttons

If you find that you’re suppressing most or all of the system-defined buttons on a form, you should reexamine your form style (Form.Design.Style) or pattern, and reconsider the purpose of the form. Should the form be a dialog instead? (By default, dialogs don’t receive any system-defined buttons.) Often, forms that are in this situation have Style=Auto and would be more appropriate as dialogs. If your form should not technically be a dialog, there is no currently no metadata or code that can automatically suppress all the system-defined buttons at the same time. Unless you switch the form to a form style that doesn’t receive any system-defined buttons, you must to suppress/override each button individually (see the other sections in this article). This scenario should be extremely rare.

New and Delete system buttons

The New and Delete buttons are currently added by the kernel, and are controlled via special metadata properties. These buttons always work on the first master data source on the form.

When are these buttons available by default?

Forms usually have system-defined New and Delete buttons. These buttons appear on a form under the following conditions:

- The form has a style that allows for system-defined buttons.
- There is at least one data source on the form.

How do I affect the visibility of the New and Delete buttons on a form, but still allow the standard New and Delete tasks to fire via keyboard shortcuts?

Use the ShowNewButton and ShowDeleteButton properties on Form.Design to control the visibility of the New and Delete buttons on a form. Note that if the data sources still let the user create and delete records, the keyboard shortcuts will continue to work even if the system buttons aren’t visible.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form.Design.ShowNewButton</td>
<td>No</td>
<td>Suppress the system-defined New button on the form.</td>
</tr>
</tbody>
</table>
How do I affect the state (enabled/disabled) of the New and Delete buttons, and the associated task behavior on a form?

To affect both the state of the New and Delete buttons and the associated task behavior on a form, use the AllowCreate and AllowDelete properties on the first master data source. Additionally, the associated keyboard shortcuts will have no effect if you use this approach to disable the New and Delete buttons.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form.Datasources.&lt;FirstMasterDatasource&gt;.AllowCreate</td>
<td>No</td>
<td>The form doesn't allow record creation.</td>
</tr>
<tr>
<td>Form.Datasources.&lt;FirstMasterDatasource&gt;.AllowCreate</td>
<td>Yes</td>
<td>The form allows record creation. The form has an enabled system-defined New button.</td>
</tr>
<tr>
<td>Form.Datasources.&lt;FirstMasterDatasource&gt;.AllowDelete</td>
<td>No</td>
<td>The form doesn't allow record deletion.</td>
</tr>
<tr>
<td>Form.Datasources.&lt;FirstMasterDatasource&gt;.AllowDelete</td>
<td>Yes</td>
<td>The form allows record deletion. The form has an enabled system-defined Delete button.</td>
</tr>
</tbody>
</table>

**NOTE**

If the form has a New record action, that button control overrides the enabled state of the system-defined New button.

How do I change the behavior of the New task (either by clicking the button or by using the keyboard shortcut)?

There are three mechanisms for changing the behavior of the New task:

- Use the New Record Action property. This property is currently available only on Form.Design, and the referenced action will be triggered for all CommandButtons that call the New command on the form, even CommandButtons that are bound to secondary collections. In the future, the New Record Action property will be placed on all container controls to provide better control over the property's scope. Note that, currently, the New Record Action property also can't be defined as a Menu Button or Drop Dialog Button.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form.Design.NewRecordAction</td>
<td>The control name from the form</td>
<td>This property overrides the New task to perform the specified action.</td>
</tr>
</tbody>
</table>

- (Recommended) Use eventing. In particular, there are pre-events and post-events for record creation and deletion, where you can put code that is meant to run before and after these actions. See the following code example.
**How do I change the behavior of the New button (but not the task)?**

Imagine that you want to replace the system-defined New button with a menu button that lets the user select among several "New" options. Therefore, you complete the following tasks:

1. Hide the system-defined New button.
2. Model your own button for the New action.

However, at this point, the system-behavior for New still fires when the keyboard shortcut is invoked. If you want to make this modeled New button fire when the keyboard shortcut is pressed, you must set this button as the NewRecordAction on Form.Design. As we noted in the previous section, the action will currently be fired for all CommandButtons that call the New task on the form. Therefore, you should not use this approach on forms that have multiple data sources.

**Edit, Done, Save, and Restore system buttons**

The **Edit**, **Done**, **Save**, and **Restore** buttons let users switch the edit mode of the form as they require. Because this capability is critical, these buttons can't currently be suppressed. However, if a form should always be in **Edit** mode or should always be in **View** mode, you can use the **ViewEditMode** property to achieve that effect, and the buttons won't appear on the form. Note that most forms should be left as **ViewEditMode**=Auto. Don't set **ViewEditMode**=View or **ViewEditMode**=Edit for purely cosmetic reasons, but only if the form will always be read-only or will always be editable.
<table>
<thead>
<tr>
<th>FORM DESIGN VIEW EDIT MODE</th>
<th>FORM BEHAVIOR</th>
<th>SYSTEM-DEFINED BUTTON BEHAVIOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit</td>
<td>The form is always in Edit mode.</td>
<td>Only the Save and Revert buttons are shown, because the user can’t exit Edit mode. The Revert button is on the framework-provided Options tab.</td>
</tr>
<tr>
<td>Auto</td>
<td>The form can be switched between View mode and Edit mode.</td>
<td>In View mode, the Edit button is shown on the Action Pane. In Edit mode, the Save button is shown on the Action Pane, and the Read mode and Revert buttons are shown on the framework-provided Options tab.</td>
</tr>
</tbody>
</table>

Can I conditionally suppress or show the Edit button?

If a form is editable at some points and read-only at other points, the logic that is used to determine whether the user can edit the form can be used to set the ViewEditMode property at run time. When the form should be read-only, set ViewEditMode=View. When the form can be either edited or viewed, set ViewEditMode=Auto.

How do I run additional code when the Edit button is clicked?

To have additional code run when the user clicks the Edit button, use eventing (see the example in the "How do I change the behavior of the New task (either by clicking the button or by using the keyboard shortcut)?" section earlier in this article). Specifically, use the following events:

- To subscribe to view/edit mode switching, use these events:
  - element.viewEditModeHelper().EditModeSwitching
  - element.viewEditModeHelper().EditModeSwitching
- To query the current view/edit mode, use these events:
  - element.viewEditModeHelper().isInEditMode()
  - element.viewEditModeHelper().IsInViewMode()
- To trigger view/edit mode switching, use these events:
  - element.viewEditModeHelper().setViewEditMode(ViewEditMode::Edit)

Refresh, Popout, and Close system buttons

The Refresh, Popout, and Close buttons are system buttons that are located on the right side of the Action Pane. The Refresh button is used to refresh all data on the form. The Popout button is used to move the current form into a separate window. The Close button closes the form (essentially, this button clicks the browser’s Back button). These system buttons are integral components and can’t currently be suppressed.

Other system-defined buttons

The remaining system-defined buttons are added during Form.Init. They are added only if a control of the same name doesn’t already exist.

How do I run additional code together with a system-defined button or change the behavior of the button?

- For view switching buttons (for example, buttons that switch to the grid view, details view, header view, or lines view), you should override the pageActivated() method on the individual TabPage.

- For system-defined buttons that have pre-eventing/post-eventing (for example, New, Delete, and Edit), you can subscribe to the appropriate events. See the corresponding sections for New/Delete and Edit buttons for information about specific events for those buttons.
How do I suppress any of these system-defined buttons on a form, but without suppressing any corresponding task?

In general, we don't recommend that you suppress any of the system buttons. Their presence provides consistency to actions on forms. However, the buttons currently appear in some situations where they aren't as useful. We have future work items to address many of these situations. For example, have work items for the following tasks:

- Suppress the Attach button on forms that aren't configured to allow attachments.
- Suppress the Export button on forms that have no grids.
- Suppress the Show filters button on forms that don't have a main grid.

However, if you must suppress one of these buttons (strongly discouraged), you can find the control via code and set its visibility to false, as shown in the following code example. Use SysSystemDefinedButtons macros, where they are available, to reference the button names.

```java
public void init()
{
    //SysSystemDefinedButtons
    super();
    attachButton= this.control(this.controlId(#SystemDefinedAttachButton)) as FormCommandButtonControl;
    attachButton.visible(false);
}
```
This topic describes the steps for displaying images on a page or in a grid. The topic also provides background about some of the ways that images can be used, and the APIs that are used.

**NOTE**  
For accessibility, when you use an image to indicate status or show data, the image must be accompanied by a tooltip, enhanced preview, label, or other textual representation that describes the value or status that the image represents.

Finance and Operation apps do not use embedded resources for images. Instead, it uses lightweight symbols. The coding pattern has changed slightly to support the new image control.

For ImageList uses, the runtime accepts the old `ImageID` value and maps it to a symbol, so that existing code continues to work.

**NOTE**  
In some cases, there is no image even after runtime mapping, and this behavior is intentional.

AX 2012 displays images in a grid column to indicate status. These images were sometimes retrieved from embedded resources that are no longer available.

AX 2012 offers the following storage options for images:

- An embedded resource where images are offered as part of the kernel itself
- An Application Object Server (AOS) resource where developers or independent software vendors (ISVs) can add their own image resources
- A file location where developers or ISVs can load images at run time
- A database field that is stored as a bitmap

The following storage options are available for images:

- An AOS resource where developers or ISVs can add their own image resources
- A URL location where developers or ISVs can load images at run time
- A database field that is stored as a container.
- A symbol font, where images are rendered by name from the font

Images that are stored as AOS resources allow for the use of an image that isn’t categorized as user data, and can be used with your application.

**NOTE**  
If there are legacy embedded resource images that UX has approved for use, those embedded images can be manually transferred to an AOS resource and used.

A typical web application maintains a collection of images on an Internet Information Services (IIS) server and just provides a URL to the image. Although this approach is supported, we don't expect that it will be used very much. Instead, we expect that the symbol font will be used as an image source.
Of course, application logic will store an image in a database to allow for strong employee photos, product images, and so on, and this approach is a first-class experience.

A symbol font is the most performant and scalable image format. We expect that characters from the symbol font will be used for most application use cases (grid row by row status, button images, and so on).

For the list of symbols that are available in the symbol font, see Symbol font.

**Image type: Symbol**

<table>
<thead>
<tr>
<th>PROS</th>
<th>CONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Usually, the symbol font is the smallest payload to send to the client.</td>
<td>A limited number of framework-defined symbols is available.</td>
</tr>
<tr>
<td>• You can easily customize the images by using Cascading Style Sheets (CSS).</td>
<td></td>
</tr>
<tr>
<td>• The symbol font should already be cached on the user's computer. Therefore, no extra bandwidth is used, and there are no additional network requests that might slow down page loads.</td>
<td></td>
</tr>
<tr>
<td>• Colors can be controlled by themes.</td>
<td></td>
</tr>
<tr>
<td>• The images are automatically scaled on high-DPI displays.</td>
<td></td>
</tr>
</tbody>
</table>

**Design time**

**Image location:** Symbol  **Typical image:** “Person”

**Run time**

Sometimes, you don’t have an image for a particular record in a grid, but you don’t want an empty space where the image should be. The following example shows how you can use a display method to check for an image value, and then substitute a placeholder image instead.

```java
public display container customerImage()
{
    ImageReference imgRef;
    container imgContainer = this.Image;
    if(imgContainer == connull())
    {
        // there is no image... the container is null
        // show a generic person outline image
        imgRef = ImageReference::constructForSymbol("Person");
        imgContainer = imgRef.pack();
    }
    return imgContainer;
}
```

**Image type: AOT Resource**

<table>
<thead>
<tr>
<th>PROS</th>
<th>CONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>In addition to the pros of using URL images (see the next section), AOT resources are modeled and managed by the development tools.</td>
<td>A limited number of framework-defined images is available.</td>
</tr>
</tbody>
</table>

**Design time**

You just create a new resource and then save the image into the Application Object Tree (AOT) resource. When
you model your image control on a page, you specify the resource name, not the image name. This approach is typically used for legacy images (icons) that don’t have equivalents in the symbol font. Image location:
AOTResource Typical image: “ResourceMicrosoft Dynamics AX” (a .jpg is added to resources) | Run time

```java
public display container imageDataMethod()
{
    ImageReference imgClass = ImageReference::constructForAotResource(
        "ResourceMicrosoft Dynamics AX");
    return imgClass.pack();
}
```

Image type: URL Image

<table>
<thead>
<tr>
<th>PROS</th>
<th>CONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>• This approach provides an easy way to reference any image anywhere on web.</td>
<td>• The transfer size isn’t as small as it is for symbols, but it’s reasonable. The URL is sent as a string for each control that uses the image. The browser then downloads the image from the URL, and from that point, standard browser caching rules apply.</td>
</tr>
<tr>
<td>• This approach supports full-color images.</td>
<td>• You can’t easily theme the images by using CSS.</td>
</tr>
<tr>
<td>• The web browser can cache the image, based on the settings of the server that hosts the image.</td>
<td>• Unless the URL points to a Scalable Vector Graphics (SVG) file, the image isn’t automatically scaled on high-DPI displays.</td>
</tr>
</tbody>
</table>

Run time

The following example shows an image that uses a URL that is contained in a string.

```java
public display container imageDataMethod()
{
    ImageReference imgClass = ImageReference::constructForUrl(this.ImageURL);
    return imgClass.pack();
}
```

This code sends a small JavaScript Object Notation (JSON) message to the control on the client. This message instructs the control to treat the image as a URL and let the browser do the work of downloading the image. No download occurs on the server. Storing an image URL in a database table You can also have a container field for the image column on your table. You can then use code that resembles the following example to store the `ImageReference` pack.

```java
ImageReference imgClass;
CLIControls_ImageTable imgTable;
ttsbegin;
    imgClass = ImageReference::constructForUrl(
        "http://dynamics/PublishingImages/ERPlogs/DynamicsLogo.jpg");
    imgTable.ImageField = imgClass.pack();
    imgTable.insert();
ttscommit;
```

This code causes the user’s browser to download the image from the specified URL. The use of `ImageReference` involves some overhead, but this approach lets you use a single application programming interface (API) to handle images that are created from binary data, URLs, AOT resources, or symbols. You can even mix and match image types between rows of data.
## Image type: Binary Image

<table>
<thead>
<tr>
<th>PROS</th>
<th>CONS</th>
</tr>
</thead>
</table>
| Usually, this approach offers the easiest migration if the Image class in X++ was already used, or if binary images were previously stored in the database. | • This approach involves the largest transfer size, because the binary image is encoded as a string and sent to the client as part of the interaction.  
• The browser can’t cache the images.  
• For a grid, the binary-encoded image is sent for every row, even if multiple rows use the same image. Therefore, this approach can lead to very large transfer sizes in the interactions.  
• You can’t easily theme the images by using CSS.  
• The images aren’t automatically scaled on high-DPI displays. |

### DESIGN TIME | RUN TIME

**Using a database field** This approach is typically used to display data, such as employee pictures and product images. You can bind directly to a field, or you can use a display method. Data Source Data Field Data Method

Typically, the images are loaded from database, and no additional code is required. For cases where the image is managed in a data method, see the data method examples.

## Display methods and images (three return types)

When you use a display method for an image type to show an image in a grid, three return types are understood by the image control that works with the framework. All three return types can be used to display an image.

- Int (imagelist array index)
- Container (image instance)
- ResID (which is mapped to a symbol)

### NOTE

ResID and Int are the same return types. If the `imagelist` property of the image control instance has been assigned an instance value, the display method return value is considered an array index into the imagelist. If the `imagelist` property is `null`, the return value is used to map a legacy ResID to a symbol.

## Images in a grid and the legacy ImageList collection

In AX 2012 and earlier versions, a common use pattern for displaying images is to store an image as a resource or use a kernel-supplied image resource, and then at run time, extract that image and place it in a reusable collection that is known as an ImageList. The guidance is to use lighter-weight symbol images. You should rewrite all legacy code so that it uses symbols directly. You should also replace all code that uses the ImageList collection. If you don’t make these changes, the legacy ImageList collection won’t display images, because use of this collection relies on embedded (kernel) resources that no longer exist. Therefore, to support legacy code until it can be updated, the ImageList collection maps the ResID for an embedded resource to a new font-based symbol to help guarantee that any code that uses the ImageList collection will continue to run and provide an image.
Using the imageList property for backward compatibility

An image control has a property that is named `imageList`. You pass in an instance of the ImageList collection to this property. In this way, the image is an array of images that you select via the array number.

```java
public void init()
{
    int imgCnt;

    // create an imagelist instance
    Imagelist imageList = new Imagelist(ImageList::smallIconWidth(), Imagelist::smallIconHeight());
    super();

    // add images to the instance (return value is not needed)
    // Note that a legacy ResID is used in the new Image constructor.
    // This is a compatibility mapping of resource to symbol.
    imgCnt = imagelist.add(new Image(#ImageInfo));
    imgCnt = imagelist.add(new Image(#ImageWarning));
    imgCnt = imagelist.add(new Image(#ImageError));

    // pass the imagelist instance to the control
    ImageListDM.imagelist(imageList);
}

// at runtime, select the image you want to show: when the control has an imagelist instance,
// this int value is used to index into that array
public display int imageListDataMethod()
{
    int imgCnt = imageCnt mod 3;
    imageCnt++;
    return imgCnt;
}

/*
Note: The legacy image resource ID's #ImageInfo, #ImageWarning, #ImageError are
mapped from the legacy resource id to a symbol name in the X++
class ImageLoader
*/
```

Display method that returns an ImageRes (legacy image resource ID)
// this is an example of backward compatibility the use of ImageRes will become obsolete

display ImageRes checkIfError(HRMCompEventEmpl _hrmCompEventEmpl)
{
  if (!_hrmCompEventEmpl.RecId)
  {
    return 0;
  }
  if (_hrmCompEventEmpl.Status == HRMCompEventEmplStatus::Ignore ||
      _hrmCompEventEmpl.Status == HRMCompEventEmplStatus::Approved ||
      _hrmCompEventEmpl.Status == HRMCompEventEmplStatus::Loaded)
  {
    return 0;
  }
  else
  {
    if (_hrmCompEventEmpl.ErrorStatus == HRMCompEventErrorStatus::Error)
    {
      return ImageError;
    }
    if (_hrmCompEventEmpl.ErrorStatus == HRMCompEventErrorStatus::Warning)
    {
      return ImageWarning;
    }
    if (_hrmCompEventEmpl.ErrorStatus == HRMCompEventErrorStatus::Info)
    {
      return ImageInfo;
    }
    return 0;
  }
}

Display method that returns a container

public display container checkIfError(HRMCompEventEmpl _hrmCompEventEmpl)
{
  ImageReference  imageReference;
  container       imageContainer;
  if (_hrmCompEventEmpl.RecId && _hrmCompEventEmpl.Status == HRMCompEventEmplStatus::Created)
  {
    switch (_hrmCompEventEmpl.ErrorStatus)
    {
      case HRMCompEventErrorStatus::Error:
        imageReference = ImageReference::constructForSymbol('Error');
        break;
      case HRMCompEventErrorStatus::Warning:
        imageReference = ImageReference::constructForSymbol('Warning');
        break;
      case HRMCompEventErrorStatus::Info:
        imageReference = ImageReference::constructForSymbol('Info');
        break;
    }
  }
  if (imageReference)
  {
    imageContainer = imageReference.pack();
  }
  return imageContainer;
}

Obtaining and displaying an image from the user by using file upload

Model a page that has an image control and a FileUpload button.
<!-- model a new FileUpload control (style=minimal) -->
// class declaration
FileUpload uploadControl;

// form init() create a callback event handler to be notified when upload is complete
public void init()
{
    // when uploading an image, this method is called upon completion.
    uploadControl = FileUpload1;
    uploadControl.notifyUploadCompleted += eventhandler(this.UploadCompleted);
}

// form close() release the callback event handler
public void close()
{
    // when the form closes, release the eventhandler for file upload callback
    // FileUpload uploadControl;
    super();
    // uploadControl = FileUpload1;
    uploadControl.notifyUploadCompleted -= eventhandler(this.UploadCompleted);
}

// when the upload completes, grab the image and store it in the database
/// <summary>
/// This method is called by the file upload mechanism, when the upload completes
/// </summary>
public void UploadCompleted()
{
    Binary binaryImage;
    System.IO.MemoryStream stream;
    String255 myUrl;
    if(uploadControl.uploadSuccess())
    {
        InteropPermission perm = new InteropPermission(InteropKind::ClrInterop);
        perm.assert();

        // BP Deviation Documented
        webClient = new System.Net.WebClient();

        // BP Deviation Documented
        // if success, downloadURL contains the path to the Azure blob location for the file
        stream = new System.IO.MemoryStream(webClient.DownloadData(uploadControl.downloadUrl()));

        // grab the data and assign to the image field
        binaryImage = Binary::constructFromMemoryStream(stream);

        // assign to the database field (type=container)
        FMVehicleModel.Image = binaryImage.getContainer();

        CodeAccessPermission::revertAssert();
    }
}

Example of in-memory bitmap manipulation

In this example, an image is created from scratch. However, developers can also load a bitmap from an alternative source and then manipulate the image as desired (for example, by cropping, stretching, or resizing, or by changing the opacity). After any manipulation is completed, the developers can display the image by using the image control, or they can assign it to a data source field.
public void clicked()
{
    Binary binaryImage;
    Image image;
    int x,y;
    super();
    InteropPermission perm = new InteropPermission(InteropKind::ClrInterop);
    perm.assert();
    /*
    In this example, we'll create a bitmap programmatically, we'll use a memory Stream o'bytes to then convert to the container format the image control expects.
    */
    System.Drawing.Bitmap bitmap = new System.Drawing.Bitmap(100,100);
    System.IO.MemoryStream myStream = new System.IO.MemoryStream();
    // draw some stuff (or load a bitmap from an alternative source)
    for( x=0; x < bitmap.Height; ++x)
    {
        for( y=0; y< bitmap.Width; ++y)
        {
            bitmap.SetPixel(x,y,System.Drawing.Color::White);
        }
    }
    for(x=0; x < bitmap.Height; ++x)
    { 
        bitmap.SetPixel(x,x, System.Drawing.Color::Red);
    }
    // move our bitmap to an in memory stream
    bitmap.Save(myStream, System.Drawing.Imaging.ImageFormat::Bmp);  
    // stream goes to raw binary
    binaryImage = Binary::constructFromMemoryStream(myStream);
    // create a blank image and copy our binary data to the image format
    image = new Image();
    image.setData(binaryImage.getContainer());
    // copy the image data to the image control
    MyImage.image(image);
    // alternatively, skip the image conversion step and assign directly to the data field
    binaryImage = Binary::constructFromMemoryStream(myStream);
    // assign to the database field (type=container)
    datafield.Image = binaryImage.getContainer();
    CodeAccessPermission::revertAssert();
}

Additional examples (URL, binary, and symbol)
The following table explains two concepts: Image Class and FormImageControl.
**Image Class**

This class is a run-time representation of an image.

Four constructors:
- `Image::ConstructBinary(INT64 Encode);`
- `Image::ConstructSymbol(SymbolName);`
- `Image::ConstructURL(URL);`
- `Image::Construct(ResourceName);`

**FormImageControl**

This control is used to add an image at run time.

.image(new image());

---

**Using a display method to show an image from a URL string**

In this example, a display method is used to translate a string that contains a URL to the format that the image control expects.

```java
public display container imageDataMethod()
{
    ImageReference imgClass = ImageReference::constructForUrl(this.ImageURL);
    return imgClass.pack();
}
```

This code sends a small JSON message to the control on the client. This message instructs the control to treat the image as a URL and let the browser do the work of downloading the image. No download occurs on the server.

**Using a display method to show a blank image**

There might be times when you have no image for a particular record in a grid, but you don’t want an empty space where the image should be. This example shows how you can use a display method to check for an image value and then substitute a placeholder image instead.

```java
public display container customerImage()
{
    ImageReference imgRef;
    container imgContainer = this.Image;
    if(imgContainer == connull()) // there is no image... the container is null
    {
        imgRef = ImageReference::constructForSymbol("Person"); // show a generic person outline image
        imgContainer = imgRef.pack();
    }
    return imgContainer;
}
```

```java
public display container statusImageDataMethod()
{
    ImageReference statusImage;
    if (this.Status == NoYes::Yes)
    {
        statusImage = ImageReference::constructForSymbol("Accept");
    }
    else
    {
        statusImage = ImageReference::constructForSymbol("Cancel");
    }
    return statusImage.pack();
}
```
Taking an image URL and storing the image in table

You can have a container field for the image column on your table. You can then use code that resembles the following example to store the `ImageReference` pack.

```plaintext
ImageReference imgClass;
CLIControls_ImageTable imgTable;
ttsbegin;
  imgClass = ImageReference::constructForUrl(
    "http://dynamics/PublishingImages/ERPLogos/DynamicsLogo.jpg");
  imgTable.ImageField = imgClass.pack();
  imgTable.insert();
ttscommit;
```

Like the display method that is described in the “Using a display method to show an image from a URL string” section, this code causes the user’s browser to download the image from the specified URL. Although this approach involves some overhead, you can use a single API to handle images that are created from binary data, URLs, AOT resources, or symbols. You can even mix and match image types between rows of data.
This topic provides information about the new color picker control that lets users select a color.

Traditionally, color has been considered an ideal way to communicate with a user. For example, the color red is often used to draw the user's attention to information that is important. However, some users can't distinguish certain colors or shades, and some users are blind. Therefore, we don't recommend that you use color alone to communicate information to the user. Instead, you should use color together with a symbol or additional text to convey information to all users.

### Color selection in Dynamics AX 2012

In Microsoft Dynamics AX 2012, color selection had these characteristics:

- It used the Win32 color picker.
- It required Win32 application programming interfaces (APIs) for RGB/decimal conversion. (The input control accepted a decimal value for RGB.)

```csharp
Public void lookup()
{
    #DEFINE.COLORVALUE(64)
    Int r,g,b
    container choosencolor;
    Binary customcolors = new Binary(#COLORVALUE);
    CCColor colorvalue;
    Super();
    [r,g,b] = WinAPI::RGBint2Con(this.backgroundColor());
    chosenColor = WinAPI::chooseColor(element.hWnd(),r,g,b, customColors, true);
    If(chosenColor)
    {
        [r, g, b] = chosencolor;
        Colorvalue = WinAPI::RGB2int(r,g,b);
        This.backgroundColor(colorValue);
        employeeWorkPlannerForm parmAbsenceColor(colorvalue);
        Employeetable.columns(employeeworkplannerform.numberofcolumns());
        Absencecolorparm = colorvalue;
    }
}
```

### Color selection for input controls

In the current version, the color picker control is a standard control type. The color picker control can be put directly in a form, or it can be used as part of a custom lookup for an integer or string control. The following example shows how to interact with the color picker control in a custom lookup. However, the code is similar if you put the color picker in a form and provide the user with a button to select a color.

- A color picker control can be hosted in a form or a custom lookup to let the user visually pick a color or specify an RGB value.
The return value is a decimal value that can be assigned directly to an input control property. (No run-time RGB conversion is required.)

```csharp
[Control("String")]
class stringControl
{
    /// <summary>
    /// </summary>
    public void lookup()
    {
        int color = hex2Int(this.valueStr());
        color = ColorSelection::selectColorStringControl(this, color);
        this.text(int2hex(color));
        this.backgroundColor(color);
    }
}

[Control("Integer")]
class integerControl
{
    /// <summary>
    /// </summary>
    public void lookup()
    {
        int color = this.value();
        color = ColorSelection::selectColor(this, color);
        this.value(color);
        this.backgroundColor(color);
    }
}
```

Using color in a table control

There is no design-time experience for coloring input controls. In other words, you can’t model an input control so that it's “blue” by default. However, there are run-time capabilities that let you change color values. The following example shows how you can change the way that the cells of a table control are colored.

```csharp
public FormControl editControl(int column, int row)
{
    stringEdit.colorScheme(FormColorScheme::RGB);
    stringEdit.backgroundColor(WinAPI::RGB2int(225,225,125));
    stringEdit.foregroundColor(WinAPI::RGB2int(8,10,200));
}
```

Using color in a grid control
public void displayOption(Common _record, FormRowDisplayOption _options)
{
    CLIparentTable table;
    table = _record;

    if(!cleared)
    {
        _options.affectedElementsByControl(CLIparentTable_AInt.id());
        _options.affectedElementsByControl(CLIparentTable_AEnum.id());
        _options.affectedElementsByControl(CLIparentTable_AString.id());
        _options.affectedElementsByControl(CLIparentTable_EditMethodString.id());

        if(table.AInt<=20)
        {
            _options.backColor(WinAPI::RGB2int(255,165,0));
        }
        else if( table.AInt>20 && table.AInt <60)
        {
            _options.backColor(WinAPI::RGB2int(255,255,0));
            _options.fontItalic(true);
            _options.textColor(WinAPI::RGB2int(255,0,127 ));
            _options.fontStrikethrough(true);
        }
        else
        {
            _options.backColor(WinAPI::RGB2int(128,0,128));
            _options.fontUnderline(true);
        }
    }
    super(_record, _options);
}

Static RGB instead of run-time conversion from integer to RGB values

Previously, run-time conversion that used WinAPI::RGB2Int was required, because the Win32 color picker returned an RGB value, whereas the background color APIs accepted an integer. This run-time conversion isn’t required, because the new color picker returns an integer to match the control’s consumption of an integer. Additionally, it’s understood that .NET code often uses RGB values for colors. Therefore, in those cases, run-time conversion of colors isn’t required for each use. Instead, you can define static color variables. Here are three examples.

Static int GrayColor = 220 + 220 <<#offset8 + 220<<offset16;
Static int GrayColor = 0xdcddcd
Static int GrayColor = 14474460; // DCDCDC or 220,220,220
In the area of right-to-left (RTL) language support, one consideration is the combination of RTL text and left-to-right (LTR) text in the same string. This topic discusses the issue of bidirectional text and how it's handled.

A great example of right-to-left language support: Microsoft Word

In the area of right-to-left (RTL) language support, one consideration is the combination of RTL text and left-to-right (LTR) text in the same string. One example of a program that implements this functionality correctly is Microsoft Word. If you're trying to understand the correct behavior of mixed language presentation, you can use Word for validation. The problem is that most software just implements the Unicode standard to display bidirectional data, without evaluating how that data is actually used. Additionally, there's no attempt to provide the interactive experience that the user actually requires.

To understand how Word “gets it right” and provides a great experience, you can inspect the XML of a Word document. There, you will see that Word tracks (and stores together with the run of characters) the keyboard that is used to enter each character, and that it treats each character as a member of the language that is associated with the keyboard. Therefore, the character is given the behavioral aspects of that language.

Keeping track of character orientation in a financial program that might record billions of transactions and multi-billions of characters would produce significant transnational and spatial overhead if we stored contextual information for each character. Therefore, this behavior would be considered only for special conditions.

Bidirectional text

To support Arabic and Hebrew, both of which are RTL languages, there is an RTL orientation for the controls in each form, so that an RTL reader can interact with the form in a natural reading manner. For the most part, RTL orientation of the controls works as expected and provides RTL users with the experience that they expect. Finance and Operations apps and modern browsers support RTL orientation, and Finance and Operations app conform to that functionality. However, in some cases, extensible controls (custom controls) require special code to orient their elements correctly.

A point of reference in this article is the Win32 CEdit control, which is used primarily for standard text entry (account name, description, user name, and so on, in Microsoft Dynamics AX 2012). The behavior of the HTML Input control mimics the functionality of the CEdit control. Therefore, the same behavior applies to Finance and Operations.

The CEdit control is a Win32 control that is governed by the rules for bidirectional text management that are defined by the Unicode standard. Bidirectional text occurs when the control hosts both RTL text (such as Arabic or Hebrew) and LTR text within the same string of characters.

When you evaluate the examples in this article, remember that, regardless of the orientation of the form (RTL or LTR), the actual text that is presented is never reversed or “mirrored.” English text is always read LTR and Arabic/Hebrew text is always read RTL. When LTR and RTL text are combined, the reader must jump to the beginning of the run of characters in a given orientation. For example, when a mixed string is read from right to left, the individual words might be read like this:

-----> <------- ------- > <--------
English and Arabic/Hebrew text together: Bidirectional issues

The visual presentation (glyphs) of English, Arabic, and Hebrew characters on the corresponding keyboards clearly differ. However, those three keyboards also share some symbols. These symbols include numerals, and formatting characters such as parentheses, brackets, and underscores. According to the Unicode bidirectional algorithm, when these characters are used in a bidirectional string, their RTL/LTR orientation depends on the context of the characters that surround them. From the Unicode standard:

Note: You can find the Unicode display algorithm at https://www.unicode.org/reports/tr9/. (Section 3.3.4 of the algorithm describes how to position neutrals.)

- Characters that have a weak bidirectional type determine their directionality according to their proximity to other characters that have strong directionality.
- Characters that have a neutral bidirectional type determine their directionality from either the surrounding strong text or the embedding level.

The following table describes the bidirectional character types.

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
<th>GENERAL SCOPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong</td>
<td>L</td>
<td>Left-to-Right</td>
<td>LRM, most alphabetic, syllabic, Han ideographs, non-European or non-Arabic digits, ...</td>
</tr>
<tr>
<td>Strong</td>
<td>LRE</td>
<td>Left-to-Right Embedding</td>
<td>LRE</td>
</tr>
<tr>
<td>Strong</td>
<td>LRO</td>
<td>Left-to-Right Override</td>
<td>LRO</td>
</tr>
<tr>
<td>Strong</td>
<td>R</td>
<td>Right-to-Left</td>
<td>RLM, Hebrew alphabet, and related punctuation</td>
</tr>
<tr>
<td>Strong</td>
<td>AL</td>
<td>Right-to-Left Arabic</td>
<td>Arabic, Thaana, and Syriac alphabets, most punctuation specific to those scripts, ...</td>
</tr>
<tr>
<td>Strong</td>
<td>RLE</td>
<td>Right-to-Left Embedding</td>
<td>RLE</td>
</tr>
<tr>
<td>Strong</td>
<td>RLO</td>
<td>Right-to-Left Override</td>
<td>RLO</td>
</tr>
<tr>
<td>Weak</td>
<td>PDF</td>
<td>Pop Directional Format</td>
<td>PDF</td>
</tr>
<tr>
<td>Weak</td>
<td>EN</td>
<td>European Number</td>
<td>European digits, Eastern Arabic-Indic digits, ...</td>
</tr>
<tr>
<td>Weak</td>
<td>ES</td>
<td>European Number Separator</td>
<td>Plus sign, minus sign</td>
</tr>
</tbody>
</table>
The fundamental problem with the Unicode standard for bidirectional text is that it fails to capture the user’s intent. Therefore, the algorithm will move characters around within the same string, and put them in a location that the user didn’t specify or doesn’t want. This issue is troublesome for accounting and financial systems, because the data that users enter into the system might not match the corresponding source documents.

As we mentioned, Arabic, English, and Hebrew keyboards share some of the same characters. However, in some cases, those characters are positioned differently, depending on the keyboard that was used to type them, and/or the context of the surrounding characters and the orientation of the input control. These language-neutral characters include commas, periods, parentheses, hyphens, and underscore characters.

In some cases, the rules for displaying the same characters varies between languages. Additionally, those rules can change, depending on the kind of data that is displayed. For more information about this issue, see the "Issue: The hyphen used together with numbers: Language-specific behavior" section, later in this article.

Some people expect that characters that are entered in RTL mode will appear the same when the form is viewed...
in LTR mode. In other words, the expectation is that a customer can have some users who use Hebrew and others who use English on the same installation or in the same company.

**Issue: The underscore character used together with numbers**

**Description:** 123_456 appears as 456_123, although the user wants it to appear as 123_456.

**Example:** The user wants to enter an item number (such as 123_456) or a journal name (such as BA_Chk_Rev) that includes underscore characters for grouping purposes.

There is a difference between the Unicode standard and what our users want to see in a financial program. Even Word presents 123_456 as 456_123 for both Arabic and Hebrew. This behavior occurs because the underscore character is a grouping mechanism. It splits the number into groups of numbers that can be read from right to left.

Numbers are read from left to right in Arabic and Hebrew. "Item" numbers, regardless of the combination (RTL_LTR_RTL, LTR_RTL_LTR, Neutral_RTL, and so), should appear exactly the same in a paragraph of any direction or alignment. This issue isn’t easy to resolve for plain text programs. All the customer knows is that the physical item number is (from left to right) 123_456, and that the string should appear as 123_456 in every language, so that the number that users see always matches what they know the physical number is.

**Control behavior:** None of the off-the-shelf controls provide the desired behavior. Word fails too.

<table>
<thead>
<tr>
<th>WPF RICHTXT</th>
<th>WIN32 CEDIT</th>
<th>WIN32 RICHTXT</th>
<th>WORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**Workarounds:**

- When you’re using an English keyboard, use a different delimiter. For example, enter **123.456** or **123/456**.
- When you’re using a Hebrew keyboard, use a different delimiter. For example, enter **123.456**. The slash character (/) on a Hebrew keyboard produces a period (.)

**Recommendation:** None of the off-the-shelf controls provide the requested behavior. One alternative is to identify fields that must allow for this directional formatting and flag those fields as LTR for data input (right-aligned for display purposes). The program can’t automatically determine that a field must have this behavior. Therefore, if we expose an RTL/LTR flag, a customizer can modify targeted fields for the desired behavior. Although this approach enables this specific scenario, it’s important to understand that, if you extend the scenario by using characters in a combination of RTL and LTR languages, you will introduce other issues. Another alternative is to educate users about the fundamental behavior when underscore characters and numbers are used together. When underscore characters and numbers are required, users can then use a workaround to obtain the desired display behavior.

**NOTE**
The underscore character doesn’t present an issue when you combine English and RTL languages (pattern: RTL_LTR_RTL). Therefore, if you force a control to LTR when the input includes numbers/text/underscore characters, the behavior won’t be as expected. Users will have to manually reposition the cursor after each use of an underscore when they type RTL text. However, the behavior will be as expected for the use of numbers/text/underscore characters.

**Hebrew:** כגשק

**Arabic:** شفلاهام

**Issue: The hyphen used together with numbers:** Language-specific
behavior

**Description:** LTR is expected for Hebrew, whereas RTL is expected for Arabic.

**Example:** Item names that include numbers Arabic and Hebrew treat the hyphen differently when it's used together with numbers. An Arabic keyboard treats the hyphen as an RTL character, whereas a Hebrew keyboard treats it as an LTR character. Therefore, similar typed strings should be presented differently, depending upon the keyboard that was used. Some readers will be familiar with this example from a meeting of interested parties: 1-2-3-a-b-c

**Arabic:** The desired behavior is correct in Dynamics AX 2012. ش-لا-ؤ-3-1-2-

**Hebrew:** The desired behavior is incorrect in Dynamics AX 2012. 1-2-3-כ-ן-ש

The Unicode standard doesn’t provide for language-specific or keyboard-specific behavior. Instead, it supplies fundamental bidirectional behavior and treats the hyphen as an RTL character. Therefore, it presents the Arabic string correctly but the Hebrew string incorrectly.

**Control behavior:** The WPF RichTxt control produces the correct/desired behavior for each language.

<table>
<thead>
<tr>
<th>WPF RICHTEXT</th>
<th>WIN32 CEDIT</th>
<th>WIN32 RICHTXT</th>
<th>WORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Workarounds for the Hebrew user:**

- Don't type hyphens between the numbers. For example, if you type 1 2 3-A-B-C on a Hebrew keyboard, it appears in RTL as C-B-A-3 2 1. You should assume that the ABC order is correct for Hebrew, which is an RTL language. The English ABC text is reversed here for demonstrative purposes.
- Use a different delimiter between the numbers. For example, the slash character (/) on a Hebrew keyboard produces a period (.).

**Recommendation:** This pattern is an issue for Hebrew users who want to use numbers or hyphens in item names. Therefore, a global solution might not be appropriate, because there are exceptions. Phone numbers, Social Security numbers, and other source document identification numbers are always read LTR.

The WPF RichTxt control provides the desired behavior according to the strict guidance. However, it isn't clear that this behavior is always the desired behavior. That is, phone numbers, US Social Security numbers, and so on, should always be read and appear in LTR order, regardless of the language orientation. The alternative is to identify fields that must enable this behavior. The program can't automatically determine that a field must have this behavior. Therefore, you might have to use a descriptive property on the control, so that users can specify "Structured Formatting." If none of these approaches can be achieved, you must educate Hebrew users about the fundamental behavior when hyphens are used together with numbers. Users can then use one of the preceding workarounds to get the desired display behavior, by omitting the hyphens between numbers.

**Hebrew example:** (Desired and correct Operations for both Arabic and the Hebrew example in Hebrew)

**Pattern:** First (בעירה) hyphen second (English) hyphen third (שלום) hyphen forth (שלום)

**Correct:** בערירה-English-שלום-שלום-שלום-Hello

**Pattern:**

1. First (English letter) hyphen second (Hebrew letter) hyphen
2. First (Hebrew letter) hyphen second (English letter) hyphen

A RTL form appears as desired:

1. ש--א
2. -א-ו-

**Exception for phone numbers:** Often, Arabic users don’t have to use hyphens in phone numbers, because international phone numbers rarely use hyphens to separate digits. Any fundamental changes in the behavior of hyphens (for example, if you introduce use of the WPF RichTxt control) will cause phone numbers to appear incorrectly for Arabic users.

Phone numbers are always read LTR and often include hyphens. Phone numbers sometimes appear correctly, when they are shown in a grid through a display method that presents the string as LTR.

Currently, the input of phone numbers by using numbers and hyphens produces the correct display, such as 701-225-2188.

There is an issue with phone numbers if you try to use a US pattern that includes parentheses.

**Arabic/Hebrew/English (desired):** (701)225-2188

**Arabic (actual):** )701(225-2188

**Hebrew (actual):** )701(225-2188

**Recommendation:** Expose an RTL flag for controls or an extended data type for phone numbers. A customizer can force the control into LTR mode. This approach will let users enter values in the order that they want.

**Issue: LTR text combined with neutral characters in RTL input**

**Description:** English text is combined with parentheses or other neutral characters.

**Example:** A company name together with the company abbreviation, such as "Dynamics (DAT)"

In a typical example, the company name is followed by the company abbreviation, which is enclosed in parentheses. In this case, "Dynamics (DAT)" is shown as "(Dynamics(DAT). This behavior occurs because the closing parenthesis isn’t surrounded by two English characters. Therefore, the parenthesis is treated as an RTL character. It’s changed to the RTL closing parenthesis and moved to the end of the string (in RTL orientation).

**Control behavior:** None of the controls provide the desired behavior.

<table>
<thead>
<tr>
<th>Control</th>
<th>WPFRICHTEXT</th>
<th>WIN32CEDIT</th>
<th>WIN32RICHTEXT</th>
<th>WORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic/Hebrew/English (desired)</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Arabic (actual)</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Hebrew (actual)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The WPF RichTxt control has a flag that tries to format text according the first character in the string. Although the algorithm should fix this issue, it doesn't.

**Workarounds:** Don't use weak or neutral characters for grouping when you use English. For example, use "Dynamics DAT".

**Recommendation:** None of the controls provide the desired behavior. You must educate users about the fundamental behavior when weak or neutral characters are used together with English text. Don't use weak or neutral characters unless English characters appear on each side.
Create icons for workspace tiles

This topic provides guidelines and recommendations for creating and assigning icons to custom workspace tiles.

The dashboard contains a set of workspace tiles to which the user has access. Each of these tiles contains an icon specific to that workspace. For out-of-the-box workspaces provided by Microsoft, the icons used on the workspace tiles generally correspond to a symbol from the Dynamics Symbol font. This topic discusses the guidelines and recommendations for creating and assigning icons to tiles for workspaces created by Microsoft Certified Partners or individual customers.

Implementation details

For workspace icons, we recommend using an AOT resource for the icon. While the out-of-the-box symbols will work, we recommend creating your own so that multiple workspaces don't use the same icons. For each workspace that needs an icon, create a new image file that adheres to the guidelines below. Note that the recommended guidance for newer versions of the product has changed.

Modeling details

When you create a workspace tile, you need to follow these guidelines:

- Add an AOTResource for each new icon.
- On the tile corresponding to the workspace, set the following properties:
  - `ImageLocation=AOTResource`
  - `NormalImage=<name of AOTResource>`

Icon creation

Guidelines for creating images for custom workspace tiles are below. The recommended dimensions for the image and icon are based on the out-of-the-box workspace icons. While images of other sizes are allowed, the size and positioning of the icon relative to the full image should be maintained regardless of the image size.

Following these recommendations ensures that your workspace icon matches the styling and size of other workspace icons and that the content of your workspace icon does not get cropped by the CSS applied to the image.

- The image file should be a PNG file with a 1:1 aspect ratio.
- The recommended minimum image size is 50 × 50 pixels (px), where the icon is contained in a square that is centered in the image. For the minimum 50 × 50 px image size, the icon should be contained in a 30 × 30 px square in the center of the image.

**NOTE**
The crispness of out-of-box workspace icons and custom workspace icons might differ at different zoom levels. The reason for this difference is that images in PNG format have a fixed resolution, whereas out-of-box workspace icons are font glyphs that scale smoothly. For better resolution at different zoom levels, consider creating larger images with the same relative dimensions. For example, create 200 × 200 px or 400 × 400 px images.

- The icon should have a **white background with transparent content**.
Example

Consider the following image/icon that is to be used for a new workspace.

This icon would be converted to an image with a **white background and transparent content** with the icon centered in a larger image canvas as shown.

To understand how this relates to the sizing recommendations, here is the workspace icon image overload with the new sizing recommendations.

Using this image on a workspace tile yields the following result on the dashboard.
This topic shows where to find documentation about the public JavaScript APIs that can be used by extensible controls.

To minimize future breaks in extensible controls, an effort has been made to differentiate between the public and non-public JavaScript application programming interfaces (APIs) available to extensible controls. Control authors should ensure they only use public APIs, as any non-public API may be removed or modified in a future release. One of the planned modifications is to prefix the names of the non-public APIs with underscores to clearly denote access level. Documentation for the full set of public APIs can be found in Extensible Controls - Public JavaScript APIs.
This article categorizes and describes all the release criteria for controls.

Introduction

Typically, when you author a new control, the primary focus is on scenario functionality and technical implementation. However, before a control can be considered ready for shipment, it should conform to a set of best practice, quality, and development release criteria, as outlined in this article.

Control criteria checklist

This checklist assumes that you're familiar with the basics of control development. The following items highlight important implementation requirements that should be met by all controls.

Basic usage ready

A control must meet these requirements to be considered a functionally compatible and complete control.

Classes

These naming conventions are best practices but aren't functional requirements of a control:

- The Runtime class is named `[Name of control]Control`.
- The Design-time class is named `Build[Name of control]Control`.
- Any control-specific Component classes are named `[Name of control][Name of component]Component`.
- Any generic Component classes are named `Build[Name of component]Component`.

Resources

These naming conventions are best practices but aren't functional requirements of a control:

- The HTML Resource is named `[Name of control]HTM`, and the physical file is named `[Name of control].htm`.
- The JavaScript Resource is named `[Name of control]JS`, and the physical file is named `[Name of control].js`.

X++ Runtime

These items apply to the "Runtime" X++ class.

- FormControlAttribute is supplied with the build class name.
- The FormControlAttribute is supplied with the template ID.
- FormControlAttribute is supplied with the resource bundle path.
- The `FormTemplateControl` class is extended (directly or through inheritance).
- A FormProperty exists for each change-tracked property (that is, each property that must be read in the client JavaScript).
- The `New` method initializes each FormProperty instance.
- The `setTemplateId` and `setResourceBundleName` inherited methods are called with the same values that are supplied in the FormControlAttribute.
- The `ApplyBuild` method is used to interpret any design-time properties, and to apply design-time properties to the run-time properties as appropriate for the control.
- A Property getter/setter method exists for each FormProperty.
- A FormPropertyAttribute is supplied on each FormProperty's getter/setter method.
- Anytype is used as the argument type for FormProperties that have FormPropertyKind::BindableValue.
- All FormProperties are specified as ReadOnly to the JS class, via the third argument to the FormPropertyAttribute.
  - This argument affects only the read/write behavior that is seen by JavaScript, not by X++.
  - We don't recommend that you allow JavaScript to write directly to properties, because every property state change should be validated. We recommend that you use FormCommands for this purpose, instead of writeable properties.
- FormCommands that allow the state of FormProperties to be changed must validate that the control is in a valid state to allow for the property change.
  - Controls that are disabled, read-only, invisible, and so on, should prevent inappropriate state changes.

**X++ Design time**
These items apply to the "Design/Build" X++ class:

- The FormDesignControlAttribute is supplied with the control common name, [Name of control], without "Control" appended at the end.
  - The name that is supplied here will appear in Microsoft Visual Studio when the control is added to a form.
- A backing field exists for each design-time property.
- A Property getter/setter exists for each design-time property.
- A FormDesignPropertyAttribute is supplied to each design-time property.
- No code outside of the Design property getters/setters should exist in this class. (In other words, there should be no new() methods, and so on.)

**HTML**
These items apply to the .htm file, which is also referred to as the resource bundle:

- External resources, such as scripts, style sheets, and other HTM files, are loaded by using HTML standard <script> and <link> tags.
- All external resource loading tags are placed above the outermost HTML element in the file, so that they are loaded and processed first.
- The template ID is supplied via the id attribute on the outermost HTML element.
- The visibility of the outermost HTML element is bound to the Visible property.
- The sizing of the outermost HTML element is bound to the sizing binding handler.
- Binding handlers are used to programmatically modify HTML. (APIs such as getElementById in the JavaScript constructor aren't used.)

**JavaScript**

- The whole JavaScript code is wrapped in an anonymous function.
- Localizable strings are stored in the Globalize culture info object.
- Localizable strings are also stored in a label file, according to the instructions in Create localizable labels.
- Default values are provided for all properties that aren't initialized inside the constructor.
- The JavaScript constructor is added to the control JavaScript namespace.
- A reference to this is stored in an object that is named self, and self is used instead of this throughout the constructor.
- The base JavaScript control behaviors are inherited.
- Default values are applied by using a framework utility function.
- Client-side properties are defined in the scope of the constructor and are added to self.
- Observable and computed properties are used only for UI-bound behaviors.
- A prototype exists and contains all static methods that are specific to the control.
- Binding handlers that are specific to the control are stored in the control's namespace (not in the global control namespace).
- The control doesn't use or load external plug-ins (Microsoft ActiveX, Flash, Java, and so on).

**Interactivity**

**CSS/LESS**
- Prefix all class names with the template ID to prevent conflicts with other controls.
- Don't use class names that are defined on other controls, because those classes can change.

**Layout and resizing**
- The control uses the Sizing API for its outermost element. This requirement helps guarantee that the framework can correctly size and arrange the outermost element of the control.
- For advanced layout scenarios for elements that are contained in the control, use CSS Flexible Boxes, which are supported by the HTML standard.

**Browser support**
- The control correctly renders and supports all intended user interaction patterns on all supported browsers:
  - Microsoft Edge/Internet Explorer 11
  - The latest version Chrome
  - The latest version of iPad/MacOS Safari

**Tab sequence**
- Make sure that the control meets the W3C standards for tab sequence.

**Globalization**

**Right-to-left languages**
- Full RTL support will arrive after RTW.

**Localizable labels**
- The control uses a label file for UI text that is used only on the client side (not used in X++). For instructions about how to create and use these labels, see Create localizable labels.

**Task Recorder compatibility**
- Basic recording support
  - For any control that accepts user input, or that a user can interact with, input/actions must be recordable by Task Recorder. For Task Recorder to record the input, the control must use the SysTaskRecorder X++ API to specify the properties that should be recorded.
- Basic task guide support
  - For any control that can be recorded by using Task Recorder, the control should have task guide support. This support includes verifying that the task guide pop-up prompt points to the correct UI elements of the control, based on the input/action that was recorded.
  - In addition, validate that, when the task guide is locked ("on-rails"), the user can interact with the expected parts of the control. For example, for a combo box control, the user should be able to open the drop-down box to select a value, in addition to typing the value directly.
- Advanced recording support
  - Support for Cut, Copy, Paste, and Validate can be evaluated on a per-control basis. There are JavaScript and X++ methods that controls can implement to enable these features.

**Threat modeling the control**
- Vulnerabilities in the logical control (X++/C++) that are related to serialization/deserialization of data types, and command/property execution, should be reviewed and fixed.
  - Serialization threats are related to the way that the control parses or interprets data types.
- Any serializer for data types (except the built-in X++ Data Contracts and primitive type serialization) must be reviewed.
- Any parser must be reviewed to verify that it doesn't allow for arbitrary code execution or other exploitation.
- Any data access queries that are executed by the control (or by any helper classes that the control uses) must be evaluated and reviewed.
- Command/property execution threats are related to the way that the logical control handles an action that the control determines to be invalid.
  - Example of a command threat: A click command is executed when the control is in a disabled state. You must make sure that the click command is handled appropriately, based on the control state.
  - Example of a property thread: A property change is executed when the control is ready-only. You must make sure that the property change is handled appropriately, based on the control state.
- You must review the use of any .NET libraries to make sure that the .NET library is also secure.
- Vulnerabilities that are related to the client-side parts of the control (HTML, JavaScript, CSS, third-party libraries) should follow general secure web development principles. Specific examples include XSS vulnerabilities.
  - Any control that renders user data as HTML/JavaScript/CSS exposes an XSS vulnerability that must have mitigations identified.
  - Any control that renders external content in an iFrame exposes an XSS vulnerability that must have mitigations identified.
  - Any control that makes calls to third-party services must have mitigations identified and requires direct review by the client team.
  - Any control that handles authentication must have mitigations identified.

Control criteria details

This section explores the control criteria in more detail.

Basic usage ready

X++ Runtime

- FormControlAttribute Each control must supply the FormControlAttribute to the class declaration. The attribute must specify the build/design-time class that accompanies the control. The attribute must also specify the HTML template ID and the physical HTML file name (the resource bundle name).
- FormTemplateControl Each control must extend FormTemplateControl to participate in the control lifecycle.
- FormProperty Each control must declare FormProperties for every statically defined property that must participate in the change tracking system. For properties that are used only on the server side, no FormProperty is required.
- New Each control must implement the New method in order for its properties to participate in the change tracking system that propagates value changes between the client and server parts of the control. Inside the New method, each FormProperty
- ApplyBuild Each control must implement the ApplyBuild method in order for the control to be initialized based on the values that are set on it at design time. This method is primarily used to copy or transform design-time values into their Runtime equivalents. However, not all design-time properties must have Runtime equivalents, and not all Runtime properties must source their initial values from design-time properties.
- Property getter/setter Each control must implement property getters/setters for every FormProperty that is used by the control. These methods should be parm methods. Therefore, the names should begin with "parm" for methods that are getters/setters, "get" for methods that are only getters, and "set" for methods
that are only setters. At a minimum, a FormPropertyAttribute must be supplied to each method, together
with a FormPropertyKind and the name of the property as it should be made accessible to the HTML and
JavaScript in the client.

**HTML**

- **Template ID** Each control must provide an HTML id attribute on the outermost HTML element of the
  control's markup. In order for the control to be loaded at run time, this ID must match the ID that is supplied
to the FormControlAttribute.

- **Scripts and style sheets** Each control must use HTML standard <script> and <link> tags to consume
  other JavaScript or CSS files. These tags should be placed at the beginning of the HTML file for the control,
  before the HTML definition element for the control. If the files that must be loaded have dependencies, make
  sure that the order of the <script> or <link> loading tags is appropriate. Tags that appear first are loaded
  first. To load JavaScript or CSS from AOT Resources, use a site root–relative path (/Resource/Scripts or
  /Resources/Styles).

- **Data binding** Each control can participate in the HTML binding framework through use of the data-dyn-
  bind attribute on HTML elements that are contained in the control. The binding attribute enables HTML
  element properties to be bound to observable or computed properties that are located in the current data
  context.

**JavaScript**

- **Script encapsulation** Each control must wrap all its JavaScript in an anonymous function. This requirement
  helps prevent the framework's global JavaScript namespace from being populated with control-specific logic.

- **Localizable strings** Each control must use the Globalization API to store any string messages that are used
  by the control's JavaScript. Therefore, the control's JavaScript should not hard-code any strings that are
  displayed in the UI. Instead, the JavaScript should reference the string messages that are stored via the
  Globalization API. To load strings in the globalization object in HTML and JavaScript, you can use the
  $dyn.label API and pass in the identifier of the label. For more information, see Create localizable labels.

- **Default values** Each control must provide default values for any properties that aren’t initialized in the
  JavaScript. Therefore, any properties for which values are passed in to the JavaScript constructor on
  initialization (that is, the FormProperties in the X++ Runtime class) must provide a default-value dictionary
  for these properties.

- **JavaScript constructor** Each control must implement a constructor in the controls namespace. This
  constructor is the first line of code that will be executed when the control is loaded in the client. After the
  constructor is completed, the constructor’s associated object, this, is passed in to the HTML as the default
  data context.

- **Inheriting base control** Each control’s constructor must “inherit” from the base JavaScript control class.
The base JavaScript control class contains behaviors that are required by each control.

- **Applying default values** Each control must use the provided framework function to apply the default
  values to the control’s properties.

- **Adding client-side properties/functions** Each control can add client-side-only properties and functions
  to the JavaScript class, in addition to the server-side FormProperties and Commands that are passed in to
  the control constructor. The pattern for adding client-side-only properties/functions is to maintain a local
  copy of the this object and add the functions/properties to the local copy. After the control constructor is
  completed, all properties, functions, FormProperties, and Commands that have been added will be available
  in the HTML as the default data context.

- **Adding observable and computed properties** Each control can add client-side-only properties that
  participate in the observability patterns of the client. An observable property is initialized by using the
  $dyn.observable([initial value]) function. A computed property is initialized by using the
  $dyn.computed([function(){}]) function. Controls should use observable/computed properties sparingly,
  because these property can significantly harm performance if they are used incorrectly.

- **Control JavaScript prototype** Each control must implement a JavaScript prototype that extends the base
  control prototype. The prototype should contain any “static” JavaScript methods (methods that require no
Interactivity

Layout and resizing

- To enable the form developer to determine the control layout and size, set the width and height that are specified by the form developer on the control by using the $dyn.layout.sizing API, as shown in the following code. This is standard code that should be applied to all HTML control templates.

```
<div id="MyControl" data-dyn-bind="
sizing: $dyn.layout.sizing($data)"
</div>
```

Task Recorder recording support

Controls must use the SysTaskRecorder X++ API to indicate which actions on the control are "recordable."

- For controls that enable values to be set via properties, `SysTaskRecorder::addPropertyUserAction` should be called when the value is being set in X++. This method call tells Task Recorder to record the setting of the property.
- A similar method exists for commands (`SysTaskRecorder::addCommandUserAction`).

For more information, see Control the text that Task Recorder generates for a control.

Task Recorder playback support

- Task Recorder uses the control’s properties and commands to play back the control. The control must make sure that the commands and properties that it instructs Task Recorder to record can also be executed by Task Recorder when the control is played back. Task Recorder will rely on the interactable names for the properties and commands. The interactable name for a method is the name that is specified in the `FormPropertyAttribute` or the `FormCommandAttribute`.

Task guide support

The task guide will ask the JavaScript part of the control for the DOM element that the task guide should point to. All controls inherit basic task guide support, where the default DOM element is the outermost element of the control.

- A control can provide finer-grained details about where the task guide should point by implementing the `getTaskGuideParams` function in the JavaScript prototype for the control. This function accepts an argument (which is frequently named `options`), and this argument has a property that is named `target`. This `target` property accepts the jQuery element that the task guide should point to.
- In addition, the argument will contain information about the action that was originally recorded for the control, such as the property/command name and any arguments. The control can react to the various properties/commands that it supports by supplying the target with the DOM element that corresponds to the property/command that the user recorded.
- For some advanced scenarios, the target can also be supplied with an observable. The control can then update this observable with various DOM elements, based on events that the control exposes. This can be done by initializing the target with the observable (which contains a DOM element), and then observing an event and updating the DOM element via code in the event handler.

Task Recorder copy/paste/validate support

For controls that need to support either Copy, Paste, or Validate (mainly for advanced X++ testing purposes), the SysTaskRecorder API exposes static methods that enable the control to inform Task Recorder when a value has been copied, pasted, or validated.
Introduction

While documentation is helpful for explaining new features, it's also important to raise awareness of these new capabilities as users encounter the feature while using the product. As a result, feature callouts are available in Platform update 26. You can use feature callouts to point out a new capability to a user and optionally provide a hyperlink for the user to learn more about the feature.

In this topic, the APIs that are used to construct feature callouts are discussed in detail.

The "Got it" button

When a feature callout is triggered, the user can simply click the **Got it** button to dismiss the popup. This saves the state of this feature callout in the personalization subsystem, which prevents that specific feature callout from being triggered again.

Resetting feature callouts

Even though the feature callout state is stored in personalization, clearing personalizations will not delete the state of all previously dismissed feature callouts. Instead, separate actions have been added to reset all feature callouts so that they fire again. These actions are located on the **Personalization** tab on the Usage data page as well as on the Manage per user tab on the Personalization page.

Disabling feature callouts

If needed, administrators can turn off feature callouts for an environment using the Feature callouts enabled option on the Client performance options page.

Implementation details

The SystemNotificationsWhatsNewManager class contains two variant APIs for triggering a feature callout.

**AddWhatsNewWithActionLink()**

Add a feature callout to a control with a "Learn more" link that is configured to open the documentation associated with the new product capability.
### Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ruleID</td>
<td>Generate a unique GUID.</td>
</tr>
<tr>
<td>title</td>
<td>Provide a (localized) title.</td>
</tr>
<tr>
<td>bodyText</td>
<td>Provide a (localized) description.</td>
</tr>
<tr>
<td>targetControl</td>
<td>Provide the name of the control you want to attach the feature callout to.</td>
</tr>
<tr>
<td>urlLink</td>
<td>Provide the URL to open in a new tab when the “Learn more” link is clicked. If a URL is not specified, then a “Learn more” link will not be displayed.</td>
</tr>
</tbody>
</table>

### AddWhatsNew()

Add a feature callout to a control without a “Learn more” link.

**Parameters**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ruleID</td>
<td>Generate a unique GUID.</td>
</tr>
<tr>
<td>title</td>
<td>Provide a (localized) title.</td>
</tr>
<tr>
<td>bodyText</td>
<td>Provide a (localized) description.</td>
</tr>
<tr>
<td>targetControl</td>
<td>Provide the name of the control you want to attach the feature callout to.</td>
</tr>
</tbody>
</table>

### Example

The following code snippet will trigger a feature callout attached to the control named `TestStringControl`.

```java
public void init()
{
    super();

    SystemNotificationsWhatsNewManager::AddWhatsNewWithActionLink(
        MyTestKey,
        "My title",
        "My description",
        TestStringControl.name(),
        "https://www.microsoft.com"
    );
}
```

### Notes

- Multiple feature callouts can be shown on a page at one time.
- Only one feature callout is allowed per control. If multiple callouts exist, the last one to get triggered will be displayed.
Dialogs

There are two dialogs that replace the existing dialog box, the Slider and the MessageBox from Dynamics AX 2012:

- Slider
- MessageBox

The following sections discuss the specific goals for each concept.

Slider

The slider, or slider dialog, is a dialog box that “slides” in on top of the active page’s content from the right edge of the screen. In the following screen shot, the slider is the white region that has the caption Start rental on the right side of the window. Notice that the area to the left of the slider is shaded to help the user understand that the page beneath the slider isn’t currently available for interaction.

After a slider opens, the user can dismiss it in two ways:

- Perform an action within the slider that causes the underlying form to dismiss itself. For example, click Cancel, or enter required information and then click OK.
- Click outside the slider in the shaded area to the left. This cancels the slider, and no further actions are performed.

A slider contains a modeled form and is used to gather information from the user. Therefore, a slider should be used in most situations where a dialog box has been used in the past. For example, a slider is typically used when the user creates a new record, as in the preceding screen shot. However, a slider should not be used for simple notifications or messages to the user. For these situations, a MessageBox should be used, as described in the next section. To model a slider, you create a form, and then set the Style property to Dialog on the Form.Design node. You then model the form elements that you require (for example, fields and buttons). The
caption is defined by `Form.Design.Caption`. To simplify the process for creating sliders, we have provided the `SysBPStyle_Dialog` form as a template for modeling slider dialogs. To use this template, copy it into a new form, and then extend it as you require.

**MessageBox**

A MessageBox is a type of dialog that is rendered as a "lightbox" on top of an existing page. A MessageBox appears as a full-width modal pop-up. The following screen shot shows an example of a MessageBox.

A MessageBox is the correct mechanism to use when you must interrupt the user to notify the user about a critical situation. For example, a MessageBox is used to display a Message center error message to the user. Because a MessageBox is modal, the user can't interact with the page beneath the MessageBox until that MessageBox has been dealt with or dismissed. In the preceding screen shot, notice that the page is obscured by the MessageBox. Additionally, the areas above and below the MessageBox are shaded to help the user understand that the page isn't currently available for interaction. Be aware that, unlike a slider, the user can't dismiss a MessageBox by clicking outside it, in the shaded areas. A MessageBox can be triggered by using either the Box application programming interface (API) or any of the methods that are described earlier for triggering the display of an error. For more information, see the Message API: Message center, message bar, message details.
This topic describes the messaging system in Finance and Operations apps, specifically in terms of the application programming interfaces (APIs) that are used to create and route messages to end users.

Introduction

A new messaging system was created for Finance and Operations apps to improve this experience. Compared to earlier versions, the messaging system for Finance and Operations apps includes the following features:

- Improved association of a message with its context (form versus global).
- Improved level of interruption (none, subtle, and interrupting).
- Improved clarity between types of messages and their use.
- The control that is used to display messages is deterministic and based on form context.

Backwards compatibility of info(), warning()/checkfailed(), and error()

The info(), warning(), and error() application programming interfaces (APIs) from earlier versions of Finance and Operations apps are still supported; however, these APIs now sit upon the framework's new messaging system. Messages are routed deterministically to the message or Action center (in a non-interrupting manner) by using the context of the API call to determine the best way to present the message to the user. In general, if the use of the API originated from a form, the message appears in a message bar on that same form. (Drop dialogs and slider dialogs are both considered forms.)

The following illustration shows info, warning/checkfailed, and error message bars that correspond to page actions, or synchronous-authored messages that come from info(), warning(), and error().

If these APIs are called from a slider dialog, but that slider dialog is closed before the message appears, the message is shown in a message bar on the slider dialog’s parent page. If that slider dialog is closed before the message appears, and there is no parent page, the message is routed to the Action center. The messaging API never fails to show a message. If an appropriate host page isn’t found, the message is sent to the Action center.

NOTE

If info(), warning()/checkfailed(), or error() is called from an asynchronous process (for example, a batch), there is no form context to consider, and the messages are sent to the Action center. (To open the Action center, click the Show messages button on the navigation bar.) The following illustration shows examples of each type of message in the Action center.
NOTE
Use the **Box()** API to express an interrupting error to the user.

**Backwards compatibility of SetPrefix()**

Finance and Operations apps also support the **SetPrefix()** API for backwards compatibility. However, in the messaging system, the results of **SetPrefix()** don't actively interrupt the user; instead, the results are collected and stored (as in previous versions), and a message bar or Action center notification is presented to the user. This notification indicates that the related task has been completed, and that it might have messages that the user should review. The "Notification of results" message actually uses the task's first call to **SetPrefix()** to frame the message. This behavior is similar to the behavior in previous versions, where the first call was the "title" of the results. In this example, "Posting Results" comes from the application's first call to **SetPrefix()**.

The user can then click **Message details** to open the new **Message details** pane.

**Message()**

The **Message** API provides some useful messaging capabilities. The **Message()** API gives you more control over the lifecycle of a message by allowing you to explicitly add and remove messages. This API can be useful when validation messages need to be removed at times other than when a save boundary has been crossed, or for displaying informational messages about aspects of the user's experience that aren't necessarily related to data validation. In this example, the message is shown when the current record is displayed.
messageId = Message::Add(MessageSeverity::Informational, "The customer is marked as inactive");

The message can then be cleared when a new record is shown on the page.

Message::Remove(messageId);

Starting in version 10.0.10 / Platform update 34, you can use the `Message::AddAction()` method to embed an action within a message. This method supports adding a single action that is associated with a display or action menu item, which is then visualized as a link button. The actions are only supported in messages that are routed to the message bar until version 10.0.16 / Platform update 40, at which time these actions can be seen in messages that are routed to the Action center or the Message details pane.

In this example, a message is triggered for a system administrator indicating a particular required batch job is not running and exposes an action to go directly to the **Batch jobs** page.

```
MenuItemMessageAction actionData = new MenuItemMessageAction();
    actionData.MenuItemName("BatchJob");
    string jsonData = FormJsonSerializer::serializeClass(actionData);
    int64 messageId = Message::AddAction(MessageSeverity::Informational, "The Test batch job is not currently running", "Go to Batch jobs", MessageActionType::DisplayMenuItem, jsonData);
```

The following messaging types are supported: `MessageSeverity::Info`, `MessageSeverity::Warning`, and `MessageSeverity::Error`. Messages that use the `Message()` API are also deterministic. They can be routed to a message bar or the Action center.

**SystemNotificationsManager()**

The `SystemNotificationsManager()` class allows you to send notifications to the Action center. This class provides the following features:

- Associating one or more actions to the notification.
- Routing a notification to a set of users, or to all the users in one or more security roles.
- Defining an expiration date for the notification.
- Tracking the state of the notification (such as, you can mark a notification as "Completed").
- Defining what rule or process is raising the notification by RuleID.

In this example, a notification is raised after an export to Excel is completed by a user. The message will be available in the Action center for the next 48 hours, after which the link to the exported file is no longer available.

**NOTE**

The `AddNotification()` API was previously used in this example. As of version 10.0.23, that API is deprecated and is replaced by the `AddSystemNotification()` API. The new API requires that you set RuleID and ExpirationDateTime.
// Set up the notification
SystemNotificationDataContract notification = new SystemNotificationDataContract();
novation.Users().value(1, curUserId());
novation.Title("Export to Excel finished");
novation.RuleId('ExcelStaticExport');
novation.Message("We finished your export from the Customers page");
novation.ExpirationDateTime(DateTimeUtil::addHours(DateTimeUtil::utcNow(), 48));

// Set up the action associated with the notification
SystemNotificationActionDataContract action = new SystemNotificationActionDataContract();
action.Message("Click to download");
action.Type(SystemNotificationActionType::AxActionMenuFunction);

SystemNotificationMenuFunctionDataContract actionData = new SystemNotificationMenuFunctionDataContract();
actionData.MenuItemName(menuItemActionStr(ExportToExcelStaticOpenFileAction));
actionData.Data(fileName);
action.Data(FormJsonSerializer::serializeClass(actionData));
notification.Actions().value(1, action);
SystemNotificationsManager::AddSystemNotification(notification);

// Additional resources
User interface development home page
This topic describes the rich, powerful messaging system in Finance and Operations apps.

A new messaging system was created for Finance and Operations apps to improve this experience. Compared to earlier versions, the messaging system for Finance and Operations apps includes the following features:

- Improved association of a message with its context (form versus global).
- Improved level of interruption (none, subtle, and interrupting).
- Improved clarity between types of messages and their use.
- The control that is used to display messages is deterministic and based on form context.

Where can messages be surfaced to users?

Messages in Finance and Operations apps are generally shown in one of these places: message bars, the Action center, or message boxes.

**Message bars – Messages for synchronous tasks on the current page**

Message bars are available on primary pages, and in drop dialogs and slider dialogs. Message bars are used primarily for data validation. They can also be used to communicate messages about the state of a page or data, such as messages that are used for date effectivity. Message bars can express **info**, **warning**, and **error** statuses. Message bars should not be used for messages that require the user's immediate attention. A message bar appears when a message is first received and must be used to communicate messages only about the current page. Messages that are sent to message bars are associated with the current page. Therefore, when the user navigates away from a page that includes message bars, those messages won't appear on the new page. However, if the user navigates back to the original page, the page's messages will once again appear. Include the following information in messages:

- The condition that generated the message.
- The result if the user continues without resolving the condition that generated the message.

**Examples**

- This customer is marked as inactive.
- Customer validation has failed.
- The transaction on voucher do not balance.

**Presentation**

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Info" /></td>
<td>This customer is marked as inactive.</td>
</tr>
<tr>
<td><img src="image" alt="Warning" /></td>
<td>Customer validation has failed.</td>
</tr>
<tr>
<td><img src="image" alt="Error" /></td>
<td>The transaction on voucher do not balance.</td>
</tr>
</tbody>
</table>

**Action center – Messages from asynchronous tasks**

The Action center is located in the navigation bar. It contains messages that don't require any immediate action by the user and aren't required for the current task to continue. Typical examples include feedback from
The Action center can hold up to 500 messages. Messages are then cycled on a first in, first out basis.

**Message boxes – Errors and immediate notifications (completed synchronous operations)**

Use message boxes to alert users about issues that require immediate attention. Because message boxes interrupt users and prevent them from continuing until the message is read and dismissed, they should be used only for messages that users can’t handle later. Include the following information in error messages that appear in message boxes:

- The error that occurred.
- The cause of the error.
- Information about how to resolve the error.

An error message should include the following two components:

- **The main instruction** – This text appears in bold.
- **The message details** – This text appears below the main instruction.

**Examples**

- You can’t delete the reference number because the reference number is used in version %1.
- You have insufficient rights to perform this export.
- The root folder for catalog import processing is not configured. Configure the root folder using the Vendor catalog import parameters form.

**Presentation**

Messages of the **error** type block the user’s interaction by overlaying the current page with a modal “light box” that contains the message.

**Should I show the user a notification, a warning, or an error?**

In earlier versions of Finance and Operations apps, the **info**, **warning** (**checkFailed**), and **error** statuses weren’t always used consistently across scenarios. A message might be reported as a warning in one scenario but as an error in another scenario. When you’re deciding which status to express, use these definitions:

- **Notification** – A notification informs the user about events that might or might not be related to the current user activity. A notification can be caused by a user action or a system event, or it can provide information from the program that might be useful. Typically, a notification doesn’t require immediate user action. You notify the user by using the **info()** application programming interface (API).

- **Warning** – A warning alerts the user about a condition that might cause an issue in the future. Specifically, a warning is used for data that is in an incorrect state. Although any attempt to use this invalid data might produce an error, the fact that the current state of the data is incorrect isn’t an error condition, and the user should only be warned about the incorrect state of the data. You express data validation issues by using the **warning()** or **checkFailed()** API.
NOTE
Because of the way that the system handles the cleanup of validation messages, when the `warning()` or `checkFailed()` API is used, the return value of `validate()` should always be set to `false`. Otherwise, warning messages might not be shown to the user.

- **Error** – An error alerts the user about a problem that has already occurred. A user action that has failed is an error condition. Errors can be non-interrupting (passive) or interrupting. In a non-interrupting error, users can perform other activities before they try to correct the issue. In an interrupting error, users can’t proceed or complete the task until they correct the error condition. You express a passive (non-interrupting) error by using the `error()` API. You express an interrupting error by using the `box::` API.

**Should this message interrupt the user?**

If a task (batch job or other operation) fails, it’s often appropriate to notify the user passively. Because the user can correct the issue and retry the operation at any time, the user doesn’t have to be notified immediately. In these cases, the `error()` API is appropriate, and the user doesn’t receive an interrupting dialog. However, in other cases, the user can’t proceed until the issue is corrected. For example, if the user tries to save a page that still has invalid data, the client interrupts the user by presenting an error dialog. In these cases where it’s more appropriate to interrupt the user by presenting a dialog, the `box::` API should be used.

**Will my message end up in a message bar or in the Action center?**

The messaging system is deterministic. In other words, the messaging system uses the context of the call to determine the best way to show the message to the user. Messages are shown either in a message bar that appears at the top of pages or in the Action center, which appears in the navigation bar. The location of the message depends on where in code the message is sent from.

In general:

- If the message is caused by a page action that is synchronous (that is, the user must wait for the result), the result is shown in a message bar on the current page. (The exception is a slider dialog that was closed immediately after the action was started. Messages for slider dialogs “bubble up” to the parent page.)
- If the message is caused by an action (for example, a batch job) that is asynchronous (disconnected) and the user can continue to perform other tasks or even navigate to another page while that action is being processed, the message is routed to the Action center.

**Messaging from asynchronous or long-running background tasks**

A (potentially) long-running task should not present a message bar to the user, because message bars at the top of a page are used to present information about the current page, not some background task that might have started hours earlier. In some cases, a user who has many background tasks running continues to navigate between pages while the tasks are being completed. Therefore, messages that are presented on the current page to notify the user about background tasks are easily overlooked or ignored. Therefore, by design, background tasks send their messages to the Action center. When a new message appears in the Action center, a notification informs the user, who might be waiting for the results of an asynchronous task.

**Messaging from dialogs and slider dialogs**

The deterministic messaging system tries to send messages to the current page. However, not every call from a
dialog or slider dialog is routed to that dialog or slider. In some cases, the messaging system sends the message to the parent page instead. This behavior can occur when the messaging system is called while the dialog or slider is being closed. In some cases, the messaging system can be called when the close process for the dialog or slider is started, but the client interrupts the close process for valid reasons. Therefore, there is a “point of no return,” after which the messaging system no longer tries to send a message to the dialog or slider, and instead sends the message to the parent page. When the user clicks the OK button on the form is entering its closing sequence, shown in the code example that follows.

```java
    closeOK()
    {  // current form
        super(); // calls close()
        // parent or message center
    }
    Close()
    {  // current form
        super(); // point of no return
        // parent or message center
    }
```

If the client calls `closeOK()` or `close()` directly, then the final result might be the page or the parent page.

**When are validation messages cleaned up?**

With the messaging system in use for Finance and Operations apps, the validation message (called using the same APIs) appears in a message bar on the page itself in a passive manner. The invalid value remains but is flagged as invalid. The user can continue to enter data and can correct the validation issue at any point before the data is saved.

When a validation issue has been corrected so that the corresponding message in the message bar is no longer valid, the messaging system removes the message. The timing of message removal depends on the level where the validation logic is defined.

- If the validation logic is defined at the control or field level, the message is removed when a valid value is entered in the control or field.
- If the validation logic is defined at the table level, the message is removed the next time that the user crosses a save boundary.

If the developer needs more control over when a message needs to be removed from the UI, the `Message()` API can be utilized. See the Messaging APIs article for more details.

**I’m migrating from an older version. How do I change my existing code to use the new messaging system?**

*In many cases, no changes are required.* The messaging framework was designed to innovate and maintain backward compatibility for many common scenarios. In some cases, the program might improve the wording of messages. Alternatively, the program might use `error()` instead of `warning()`, or `warning()` instead of `error()`, to better align with the usage guidance (warnings are for data that isn’t valid, whereas errors are for failed actions). In other cases, you might decide that messages that appear on a slider dialog are more appropriate for the parent page.

**How to create a collection of related messages?**

You use `SetPrefix()` to create collections of related messages. See the Messaging APIs for more details on `SetPrefix()`. This API is largely backward compatible but is presented in a non-interrupting manner.
window isn't opened directly; instead, the user is passively notified by either an Action center message or a message bar on the page that started the task that used the `SetPrefix()` API to group the result messages into a collection. The message severity shown to the user reflects the severity of the most critical message in the collection. For example, if the collection contains no errors or warnings, the message bar is of the `info` type.

If the collection contains one or more calls to `warning()`, the message bar is of the `warning` type.

If the collection contains one or more calls to `error()`, the message bar is of the `error` type.

The use of `SetPrefix()` is also deterministic. In other words, if you use `SetPrefix()`, and there is no page context (for example, an asynchronous batch operation), the notification of results is sent to the Action center, which isn't associated with any page.
This topic provides information that will help you select the best form pattern for the forms that you migrate.

**Introduction**

The selection of a form pattern is an important step in the process of migrating a form. A pattern that is a good fit for the target form reduces the amount of migration work that is required. By contrast, a pattern that isn’t a good fit can cause wasted time and effort. Therefore, it’s important that you do some investigation, so that you can select the best form pattern for the form that you’re migrating. Here is some guidance and tips for determining the appropriate pattern for a form:

- Investigate the form’s metadata in the form designer. Pay close attention to the following details:
  - Form name
  - Form.Design.Style
  - Control names
  - The way that the controls are organized
  - The number and names of the data sources
- Investigate the form’s visuals by running the form and looking at the way information is displayed.

**Selecting a form pattern via metadata**

**Use Form.Design.Style for guidance**

The `Form.Design.Style` property often contains the name of the pattern that was previously targeted for the form. If the `Style` property correctly matches the metadata, you can use the following table to find a pattern that is likely to be a good fit for the form.

<table>
<thead>
<tr>
<th>FORM.DESIGN.STYLE VALUE</th>
<th>CORRESPONDING PATTERN</th>
</tr>
</thead>
<tbody>
<tr>
<td>DetailsFormMaster</td>
<td>Details Master</td>
</tr>
<tr>
<td>DetailsFormTransaction</td>
<td>Details Transaction</td>
</tr>
<tr>
<td>Dialog</td>
<td>Dialog</td>
</tr>
<tr>
<td>DropDialog</td>
<td>Drop Dialog</td>
</tr>
<tr>
<td>FormPart, where there are just fields</td>
<td>Form Part FactBox Card</td>
</tr>
<tr>
<td>FormPart, where there is a grid</td>
<td>Form Part FactBox Grid</td>
</tr>
<tr>
<td>ListPage</td>
<td>List Page</td>
</tr>
<tr>
<td>Lookup</td>
<td>Lookup</td>
</tr>
<tr>
<td>SimpleList</td>
<td>Simple List</td>
</tr>
</tbody>
</table>
**FORM.DESIGN.STYLE VALUE** | **CORRESPONDING PATTERN**
---|---
SimpleListDetails, where there are 2–3 fields in the navigation list (recommended) | Simple List Details – List Grid
SimpleListDetails, where there are 4–5 fields in the navigation list | Simple List Details – Tabular Grid
SimpleListDetails, where there is a tree (rare) | Simple List Details – Tree
TableOfContents | Table of Contents
Auto, where there is an **Overview** tab, a **General** tab, and a single data source | Task Single
Auto, where there are two sets of **Overview** tabs, **General** tabs, and/or headers plus lines | Task Double
Auto, where there is focus on a single record | Simple Details
Auto, where the form name ends in “Lookup” | Lookup
Auto, where there is a single tab control and **Next/Previous** buttons | Wizard
Auto, where the form name ends in “Wizard” | Wizard
Auto, where there is just a grid and some buttons | Simple List

**When a form doesn’t match the Style property**

Sometimes, a form has an incorrect Form.Design.Style property value.

**FORM.DESIGN.STYLE VALUE** | **WHAT THE FORM MIGHT ACTUALLY BE**
---|---
DetailsFormMaster | DetailsFormTransaction, if there is lines detail, or if controls have names that contain “lines”
SimpleList | SimpleListDetails, if there is more than just a grid and some custom filter fields
SimpleListDetails | SimpleList, if there is just a grid and some custom filter fields
SimpleList | ListPage, if there are numerous FactBoxes in the Parts node, or if the form has a corresponding Details Form

**Selecting a form pattern via visuals**

Although this approach is less useful than looking at the form metadata, you can get a lot of information about a form by running and examining it. Use the form visuals as an additional data point to help you select a form pattern. Look through the screen shots of migrated forms to find a form that looks like the target form. Additionally, make sure that the description or intent of the pattern matches the description/intent of the form.

**Selecting a form pattern via the designer**
Right-click the **Design** node of the target form, select **Apply pattern**, and then click the pattern to apply.

### Form pattern reference guide

#### List of classes of top-level form patterns

<table>
<thead>
<tr>
<th>FORM PATTERN</th>
<th>WHAT IT’S USED FOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details Master (two variants)</td>
<td>A form that displays the details of a complex entity</td>
</tr>
<tr>
<td>Details Transaction</td>
<td>A form that displays the details of a complex transaction entity and its lines (for example, and order and its lines)</td>
</tr>
<tr>
<td>Dialog (six variants)</td>
<td>A form that is used as a dialog to gather a set of information</td>
</tr>
<tr>
<td>Drop Dialog (two variants)</td>
<td>A form that is used as a drop dialog to gather a small set of information to provide context for an action</td>
</tr>
<tr>
<td>FactBox (two variants)</td>
<td>A Microsoft Dynamics AX 2012 FactBox that displays information about a related record or set of records</td>
</tr>
<tr>
<td>List Page</td>
<td>A Dynamics AX 2012 List Page</td>
</tr>
<tr>
<td>Lookup (three variants)</td>
<td>A form that is used as a lookup</td>
</tr>
<tr>
<td>Simple Details (four variants)</td>
<td>A form that is focused on a single record</td>
</tr>
<tr>
<td>Simple List</td>
<td>A form that displays details for a simple entity as a grid that has fewer than 10 fields per record</td>
</tr>
<tr>
<td>Simple List &amp; Details (three variants)</td>
<td>A form that displays information about an entity of medium complexity</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>A form that displays setup information or loosely related information sets</td>
</tr>
<tr>
<td>Task (two variants)</td>
<td>A legacy form pattern that is used to display master or transaction entities</td>
</tr>
<tr>
<td>Wizard</td>
<td>A form that displays a set of tab pages to the user to gather information in a predetermined order</td>
</tr>
<tr>
<td>Operational Workspace</td>
<td>A form that is used to display an overview of an activity and is meant to be a primary means of navigation</td>
</tr>
<tr>
<td>Workspace Panorama Sections (three variants)</td>
<td>A form that is used to show content for a panorama section (via a Form Part Control) in the Operational Workspace</td>
</tr>
</tbody>
</table>

**Finding forms that currently use a particular form pattern**

For a full list of forms that are currently using a particular form pattern, generate the **Form Patterns** report from within Microsoft Visual Studio. For information on running the report, see **Form pattern add-ins**. You can filter the report in Excel to find forms that use a particular pattern.

**Form pattern visuals and descriptions**

For each form pattern class, information is provided about each variant. This information includes a short...
description and an illustration of an example form.

Details Master

Details Master [Default] This form pattern is used to display the details of a complex entity on FastTabs. It includes a grid view and a details view.

Form: CustTable

Details Master w/ Standard Tabs Use this Details Master variant when your form has a large number of FastTabs (> 15) that can be grouped into categories.

Form: HcmWorker

Details Transaction

Details Transaction Use this form pattern to show the details of a complex transaction entity and its lines (for example, an order and its lines).
Form: SalesTable

Dialog – Basic [Default] This form pattern is used to gather or show a set of information.

Form: ProjTableCreate

Dialog – Read Only Use this Dialog variant when your Dialog just displays information that can’t be edited. It has only a Close button.

Form: SalesTablePostings
Dialog – FastTabs Use this Dialog variant when your Dialog content is grouped into FastTabs.

None currently in product.

Dialog – Tabs Use this Dialog variant when your Dialog content must be grouped into tabs.

Form: CaseDetailCreate

Dialog – Double Tabs Use this Dialog variant when your Dialog content has two tabs that are stacked on top of each other.

Form: PurchTableReferences
Drop Dialog

Drop Dialog [Default] This form pattern is used to initiate actions when the number of fields is small (less than five).

Form: CustCollectionsNewActivityAction

![Drop Dialog Example](image)

Drop Dialog – Read Only Use this Drop Dialog variant when the fields in the Drop Dialog aren’t editable. No OK/Close button is modeled.

No example currently exists in the product.

FactBox

FactBox Grid Use this FactBox variant to show a child collection of related information.

Form: ContactsInfoPart

![FactBox Grid Example](image)

FactBox Card Use this FactBox variant to show a set of related fields.

Form: CustStatisticsStatistics

![FactBox Card Example](image)
List Page

The Dynamics AX 2012 list page that is just a grid that is optimized for browsing records and acting on those records.

Form: SalesTableListPage

Lookup

Lookup Basic [Default] This form pattern is used if the lookup form is a grid or tree that has optional filters or buttons at the bottom.

Form: SysLanguageLookup

Lookup w/Preview Use this Lookup variant when, in addition to the basic pattern, a preview of the current record is also shown.

Form: HcmWorkerLookup

Lookup w/Tabs Use this Lookup variant when there are multiple views of a lookup (for example, a grid view/tree view or multiple filtered lists).
Form: CaseCategoryLookup

**Panorama Section**

**Form Part Section List** Use this form pattern to show a list in a workspace section. This should be modeled as a separate form and rendered in the workspace via a Form Part Control.

**Form Part Section List - Double** Use this variant when you must also show a secondary list. This secondary list isn’t initially visible.

**Hub Part Chart** Use this variant to show a chart in a workspace section. This should be modeled as a separate form and rendered in the workspace via a Form Part Control.

Form: VendInvoiceJourCountChart

**Simple Details**

**Simple Details w/Toolbar and Fields** Use this form pattern to show fields for a single base record.

Form: AgreementLine

**Simple Details w/FastTabs** Use this Simple Details variant when the record’s information is organized into FastTabs.

Form: PlanActivityServiceDetails
**Simple Details w/Standard Tabs** Use this Simple Details variant when the record’s information is organized into regular tabs.

Form: HcmEmploymentDateManager

**Simple Details w/Panorama** Use this Simple Details variant to display a record’s information in a horizontally scrolling panorama.

Form: PdsMRCEventTracker

**Simple List**

**Simple List** This form pattern is used to maintain data for simple entities.

Form: CustGroup

**Simple List and Details**

**Simple List & Details – List Grid**[Default] This form pattern is used to maintain data for entities of medium complexity. A list grid that has 2–3 fields in the navigation list is the preferred pattern for this form style in the
current version.

Form: PaymTerm

Simple List & Details – Tabular Grid Use this Simple List & Details variant if you require more than three fields in the list part of the form.

Form: ExchangeRate

Simple List & Details – Tree Use this Simple List & Details variant if the list part of the form is a tree.

Form: FiscalCalendars

Table of Contents

Table of Contents Use this form pattern to show setup information or loosely related information sets.

Form: CustParameters
**Task**

**Task Single** This legacy form pattern is used to display entities. It should be used only for migration, not for new forms.

Form: LedgerJournalTable

![LedgerJournalTable](image1)

**Task Double** This legacy form pattern is used to display transaction entities. It should be used only for migration, not for new forms.

Form: HRMAbsenceTableHistory

![HRMAbsenceTableHistory](image2)

**Wizard**

**Wizard** This form pattern is used to display a set of page views to the user to gather information in a predetermined order.

Form: WrkCtrBulkResReqEditWizard

![WrkCtrBulkResReqEditWizard](image3)

**Workspace**

**Operational Workspace**[Default] This is the preferred, performance-enhanced variant of the Workspace pattern.

Form: FmClerkWorkspace

![FmClerkWorkspace](image4)
Subpattern reference guide

List of subpattern classes

<table>
<thead>
<tr>
<th>FORM PATTERN</th>
<th>WHAT IT’S USED FOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom Filters (two variants)</td>
<td>Containers that display QuickFilters and any other modeled custom filters</td>
</tr>
<tr>
<td>Fields (five variants)</td>
<td>Containers that primarily display individual fields</td>
</tr>
<tr>
<td>Dimension Expression Builder</td>
<td>Containers that include a Dimension Expression Builder control</td>
</tr>
<tr>
<td>Dimension Entry Control</td>
<td>Containers that include a Dimension Entry Control</td>
</tr>
<tr>
<td>List Panel</td>
<td>Containers that display two lists that users move items between</td>
</tr>
<tr>
<td>Nested Simple List and Details</td>
<td>Containers that are used to embed a simpler Simple List and Details form inside a section in a form</td>
</tr>
<tr>
<td>Toolbar and Fields</td>
<td>Containers that display actions above a set of fields</td>
</tr>
<tr>
<td>Toolbar and List (two variants)</td>
<td>Containers that display actions above 1–2 grids</td>
</tr>
<tr>
<td>Workspace-related (eight variants)</td>
<td>Containers that correspond to various sections inside an Operational Workspace</td>
</tr>
</tbody>
</table>

Finding containers that require that a subpattern be applied on a form

When a form is open in the Visual Studio designer, you can easily search for containers that must still have subpatterns applied by searching for “unspecified” in the control search box at the top of the designer (as shown in the following screen shot).
**Subpattern visuals and descriptions**

For each subpattern class, information is provided about each variant. This information includes a short description and an illustration of an example form.

**Custom Filters**

**Custom Filters** Use this form pattern when custom filters are modeled. QuickFilter isn't required.

**Form: LedgerJournalTable (TopFields)**

![Image of LedgerJournalTable (TopFields)]

**Custom and Quick Filters** Use this variant when a QuickFilter is required.

**Form: CustTable (CustomFilterGroup)**

![Image of CustTable (CustomFilterGroup)]

**Fields**

**Fields and Field Groups** Use this form pattern to get a responsive layout for containers that contain only fields.

**Form: InventLocation (LocationNames)**

![Image of InventLocation (LocationNames)]

**Tabular Fields** Use this form pattern to get a structured layout of fields. It is intended primarily for totals.

**Form: LedgerJournalTransVendPaym (Balances)**

![Image of LedgerJournalTransVendPaym (Balances)]

**Fill Text** Use this form pattern when a single input control requires full width.

**Form: FmRental (Notes)**

![Image of FmRental (Notes)]

**Horizontal Fields and Button Group** Use this form pattern when a field has an inline action.

**Form: SalesTable (GroupHeaderAddressHeaderOverview)**

![Image of SalesTable (GroupHeaderAddressHeaderOverview)]
Image Preview Use this form pattern for containers that have image controls (and optional related fields).

Form: RetailVisualProfile (Login)

Toolbar and List

Toolbar and List Use this form pattern on containers that have only actions and a grid.

Form: VendTable (TabCommunication)

Toolbar and List – Double Use this Toolbar and List variant when the containers have two grids.

Form: SalesQuickQuote (TabPageExistingItems)

Workspace Related

Section Tiles Use this variant to show a set of tiles/charts in a workspace section. This should be modeled in a tab page on the workspace form. Charts are defined by using Form Part Controls.

Form: SalesOrderProcessingWorkspace

Section Related Links Use this variant to show a set of hyperlinks in a workspace section. This should be modeled in a tab page on the workspace form.

Form: SalesOrderProcessingWorkspace
Section Tabbed List Use this variant when multiple list variants must be included. Only one is shown at a time.

Section Stacked Chart Use this variant when you must include up to two charts in an Operational Workspace.

Section PowerBI Use this variant when a Power BI section must be included.

Workspace Page Filter Group Use this form pattern to add a single filter to your workspace.

Filters and Toolbar – Stacked Use this subpattern in the Form Part Section List pattern, so that actions appear below filters.

Filters and Toolbar – Inline Use this subpattern in the Form Part Section List pattern, so that filters and actions appear on the same line.

Other

Nested Simple List & Details Use this form pattern to embed a simpler Simple List & Details form inside a tab or group.

Form: HcmJob (TaskTabPage)

List Panel Use this form pattern when users must move items back and forth between two lists.

Form: CLIControls_ListPanel (FormTabPageControl1)

Toolbar and Fields Use this form pattern on containers that have only actions and fields

Form: HcmPosition (WorkerAssignmentTabPage)

Dimension Entry Control Use this form pattern on tab pages that have only a Dimension Entry Control.

Form: CustTable (TabFinancialDimensions)
**Dimension Expression Builder** Use this form pattern on containers that include a Dimension Expression Builder control.
This topic describes the concept of form patterns and discusses the process for applying and removing patterns. A list of frequent questions are also answered in this topic.

**Dynamics AX 2012: Form styles and templates**

In Microsoft Dynamics AX 2012, several form styles were introduced and formalized. Primary data types are represented by the List Page and Details Form styles. Secondary data types are represented by the Simple List and Details Form and Simple List Form styles. In addition to these core form types, other form styles exist for supporting forms, such as Table of Contents for settings and Drop Dialog for dialog forms, and Lookup for lookup forms. Other less formal form patterns, such as Wizard, also exist. Developers who wanted to build a new form of a specific style in Dynamics AX 2012 often used the corresponding template form as a starting point. After they included form content and made any modifications that were required, developers could then run the Form Style Checker add-in to validate their form in terms of structure and property values against that form style's template form.

**Finance and Operations: Form patterns**

Form patterns (a new concept that is the evolution of the Dynamics AX 2012 form templates, style, and Form Style Checker) are now an integrated part of the form development experience. These patterns provide form structure, based on a particular style (including required and optional controls), and also provide many default control properties. In addition to top-level form patterns, subpatterns can be applied to container controls, and that provide guidance and consistency for subcontent on a form (for example, on a FastTab). Patterns have made form development easier by providing a guided experience for applying patterns to forms to guarantee that they are correct and consistent. Patterns help validate form and control structures, and also the use of controls in some places. Patterns also help guarantee that each new form that a user encounters is immediately recognizable in appearance and function. Form patterns can provide many default control properties, and these also contribute to a more guided development experience. Because patterns provide many default layout properties, they help guarantee that forms have a responsive layout. Finally, patterns also help guarantee better compatibility with upgrades. Many of the existing form styles and templates from Dynamics AX 2012 continue to be supported. However, legacy form styles and templates that aren't supported have a migration path. Because the foundational elements are built based on those legacy form styles and patterns, the transition is as easy as possible.

**Applying patterns**

Applying a pattern is a straightforward process that can modify properties on multiple containers and controls on a form. Here is the standard flow for applying patterns:

1. Acquire the target.
2. Determine the pattern.
3. Apply the pattern.
4. Handle errors.

**Acquire the target**

First, you must identify a target form and add it to your project. The Form Patterns report that is generated by using a Microsoft Visual Studio add-in can help you find forms that don’t have patterns. For information on
running the report, see Form pattern add-ins. Open the report file in Microsoft Excel, and filter to a form that has no pattern. Then, in Visual Studio, open Application Explorer, and find the form. Right-click the form, and then select Add to project. When you open the form in the designer, it should have the pattern: <unselected> designation on the design node.

Determine the pattern

Decide which pattern to apply. The available patterns include those that are based on Dynamics AX 2012 form templates, and also patterns that are designed for Finance and Operations scenarios. If you require help selecting a pattern, see the Selecting a Pattern topic. For more detailed information about specific patterns, see the individual pattern guideline documents. For more information about applying a pattern, see Select a form pattern.

Apply the pattern

You can apply a pattern in three ways:

- Using metadata
- Using visuals
- Using the designer

For more information about applying a pattern, see Select a form pattern.

Handle errors

Information about the pattern appears on the Pattern tab. To learn about the pattern structure, click the control names on the Pattern tab to navigate the pattern structure. When you save or build the form, the pattern errors appear in the error list in Visual Studio.

- Double-click an error to go to the control that the error was reported for, if the control exists.
- If a control is missing, follow one of these steps:
  - If the control already exists on the form but is in a different place, move the control to the correct place, as indicated by the pattern.
  - If the control doesn't exist, create the control.

Subpatterns

After you apply a pattern to a form, you might have to apply subpatterns to the form's container controls. The process is similar to applying a pattern to a form: acquire the target, determine the subpattern, apply the subpattern, and handle any errors. To find container controls on the form that require subpatterns, search for "unspecified" in the search box at the top of the form designer in Visual Studio. These controls should have the pattern: <select> designation in the form designer. For each container, you should examine the contents and select the most appropriate subpattern. Like form patterns, the available subpatterns cover common container layouts from Dynamics AX 2012 but also include several new subpatterns. If you require help selecting a subpattern, see Selecting a Pattern. For more detailed information about specific subpatterns, see the individual subpattern guideline documents. After you've selected a subpattern, right-click the control in the form designer, select Apply pattern, and then click the subpattern to apply.

Frequently asked questions

What does applying a pattern do?

By applying a pattern, you can change multiple properties on multiple nodes in one quick action. Therefore, it's important that you understand what is happening.

Where do I find information about a pattern?

The Patterns Information panel (the Patterns tab under the form designer), the error list (View > Error List), and the form statistics add-in (right-click the form name in the form designer, and then select Add-ins > Form Statistics) all provide valuable information when you're trying to apply patterns. When you apply patterns, it's
important that you have patience, read, and proceed at a steady pace. Pattern guideline documents are also available for each pattern and subpattern. These documents contain a lot of additional information, such as information about when to use a particular pattern, what is included in the pattern, and UX guidelines to beware of when you use a pattern.

**What do I do if I make a mistake when I am applying a pattern?**

If you make a mistake, there are several actions that you can take:

- **Undo** – The Undo command (Ctrl+Z) is usually available for all actions, even for applying and removing a pattern.
- **Remove the pattern** – If you applied the wrong pattern, remove the pattern by right-clicking and then selecting **Remove pattern**. Note that properties are applied to nodes after a pattern is successfully applied without errors. Therefore, even after a pattern is removed, any properties that were changed by the pattern will still be set to the new values.
- **Revert** – When all else fails, take advantage of the source control system, and revert the changes that were made to a form.

**Why are some properties hidden when the pattern has been applied?**

Patterns enforce properties after the pattern structure has been successfully met. A property that is controlled by the pattern is hidden from developers because developers don’t have to worry about that detail. This makes the development experience cleaner by reducing the “noise” in the Property Pane from properties whose values have been set by the pattern and thus cannot be modified by a developer while the pattern is applied. Developers who are interested in the properties that a pattern is setting can remove the pattern. All the property values will then be visible on all the controls that are covered by the pattern.

**How do I identify the set of forms that I should be doing more pattern work on?**

To identify the set of forms that still have remaining patterns work, you should generate and consult the Form Patterns report. For information on running the report, see Form pattern add-ins.

- **Filter the "Pattern" column to only show "(Blanks)"** - This will show all the forms with no form pattern applied (no pattern specified on Form.Design).
- **Filter the "Unspecific count" column to only those values "greater than 0"** - This will show all forms where a pattern is expected on either Form.Design or a container control somewhere on the form. You can combine this filter with the previously mentioned filter on the "Pattern" column to show only forms with subpattern work remaining.

Note that there are no remaining patterns work left in your models if there are no rows after the filters described previously are applied. If you want to make sure that all your forms are fully covered by patterns (meaning no unspecified nodes and no Custom nodes), filter the report down to those rows that have "Percent covered controls" less than 100 percent.

**How do I find places in a form where a pattern can be applied?**

To find container controls in the form that must still have a pattern applied, search for “unspecified” in the form designer. This search will highlight all the nodes in the form that have the Pattern: `<unspecified>` designation. You can then examine each container individually to apply the most appropriate subpattern.
How do I check whether more pattern work must still be done on a form?

To determine whether more pattern work must be done on an open form, right-click the form in the designer, and then select Addins > Form statistics. If the **Pattern=Unspecified** count is more than 0 (zero), the form still has containers that must have a pattern applied. Ideally, every form should also have a **Pattern=Custom** count of 0 (zero) and a **Pattern coverage** value of 100 percent. These values indicate that the form is fully covered by patterns.

Why isn’t static text allowed inside the Fields and Field Groups subpattern? Isn’t static text allowed in forms any longer? What do I do with the static text that the Fields and Field Groups subpattern doesn’t accept? How can I show user Help instead of static text?

Static text in a form is often used as a highly visible mechanism for providing form help. Many patterns, such as the Toolbar and List subpattern and the Fields and Field Groups subpattern, don’t allow static text to be placed directly on a form. Although user assistance is a good idea, it can often be provided in other ways. At a high level, the goal is to provide better labels and a more understandable user interface, so that explicit Help content isn’t required. However, when explicit Help content is required, it should be provided as field HelpText or form-level Help content. If the static text explains the meaning of an image in a grid, consider using a tooltip to provide assistance when the user hovers over (for a mouse) or touches and holds (for touch) that image. Follow these steps to deal with static text on the form:

1. Determine a replacement for the static text on the form:
   - Consider whether the user information that is provided via static text is still required.
   - Consider whether field labels can be made more descriptive.
Both the HeightMode and WidthMode properties are not available on controls inside the Fields and Field Groups subpattern because that subpattern intentionally sets the HeightMode and WidthMode properties to be SizeToContent on input controls. An input control with a SizeToContent width is sized based on a mapping of DisplayLength to one of four pre-defined discrete sizes (extra small, small, medium, or large). This discretization of input control widths was done in an attempt to provide a fresh, clean user experience that is simple and consistent (minimizing the jagged edges caused by arbitrarily wide fields). The discrete sizes were also chosen in a way that allows these fields to be combined together to form organized and visually appealing sets of fields (since the larger field sizes are multiples of the smallest field size in terms of width).

In general, there are two other width options available for fields that are not currently allowed in the Fields and Field Groups subpattern:

- **SizeToAvailable** - We cannot allow SizeToAvailable width controls inside of this subpattern because this sizing option does not work with the ColumnsMode=Fill layout algorithm (nor does it make sense when trying to decide how to best lay out controls into columns).
- **Manual** - Controls with manual widths should be rare; the vast majority of controls should be SizeToAvailable or SizeToContent. Manual-width controls introduce inconsistency into the discrete set of field widths and column widths, especially given the fact that these controls do not adapt their size based on the user-selected density. Manual-width controls may also present difficulties in ensuring a responsive design, as they may be set to sizes that are larger than can be accommodated by a particular viewport. In an effort to preserve the desired responsive field layout and clean interface and since manually-sized controls should be extremely rare, we have opted to not allow them in the Fields and Field Groups subpattern.

Your current options for scenarios where a manually sized control is needed include using a Custom pattern (which is reasonable given the field requires a “custom” size) as well as potentially using the Fill Text subpattern for a wider field (which allows a single full-width field per container, though we plan to extend the Fill Text subpattern to allow an arbitrary number of full-width fields).

### Why do I have “unmatched” groups when I try to apply the Fields and Field Groups subpattern?

Groups and controls appear as “unmatched” in the Pattern Information panel if they, or any controls inside them, aren’t allowed by the pattern. There are two typical reasons why groups appear as “unmatched” in the Fields and Field Groups subpattern:

- There is more than one level of group depth. **Solution:** Refactor the groups so that they have only one level of depth inside the container that you’re trying to apply the Fields and Field Groups subpattern on.
- There is an image or static text inside the group. **Solution:** Remove or relocate that control, if you can.
What do I do if my form is close to a form pattern but deviates in some way that makes it Custom?

Some forms are structurally close to a defined form pattern, but because some aspects don’t fit the pattern, a Custom pattern is applied to the form. In this case, you might still be able to get some benefits of the form pattern (for example, the layout properties can be set automatically) by following these steps.

1. Modify your form so that it fully fits the pattern (for example, move or temporarily remove any controls that deviate from the pattern).
2. Apply the desired pattern.
3. Save the form, so that the property values are set by the pattern.
4. Remove the pattern.
5. Move the controls that deviate from the pattern back to their original location.
The tools for Visual Studio include a number of add-ins that support pattern usage.

**Form statistics add-in**

The *Form statistics* add-in provides a summary of the pattern usage for forms. When you access the *Form statistics* add-in from the *Dynamics 365* menu, it displays statistics for all forms. When you access the add-in from the shortcut menu for a form that is open in the form designer, it displays statistics for that form only.

![Image of Form statistics add-in](image)

**Forms Pattern report**

The *Form Patterns* report provides pattern information about every form, including whether the form uses a top-level form pattern, is a custom form, or is not specifying a form pattern. To generate the *Form Patterns* report, start Microsoft Visual Studio, click the *DYNAMICS 365* menu, expand *Add-ins*, and then click *Run the form patterns report*. The process will take several seconds. After the report has been generated, a dialog box will provide the location of the report. Browse to the specified location, and open the file in Microsoft Excel. You can then filter the report down to the models that interest you.

For more information about Visual Studio add-ins, see [Tools add-ins for Visual Studio](Tools add-ins for Visual Studio).
This topic contains the guidelines that apply to all forms, regardless of form pattern. This checklist must be used in addition to any pattern-specific guidelines.

Verification checklist

The verification checklist shows the steps for manually verifying that the form complies with the UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

Standard form guidelines

Specific form patterns and subpatterns might have exceptions to these guidelines.

- The form layout is responsive when the browser is resized or the app is run on different device sizes. In other words, all the fields should be accessible to the user either by reflowing the layout or by scrolling to the fields.

- Make sure that the form’s default View/Edit state is correct. By default, forms are in View mode. If a form should always be in Edit mode, you must explicitly set `Form.Design.ViewEditMode = Edit`.
  - The View/Edit state should be appropriate to the state of the entity. For example, if the state of the entity is Posted, and the form can’t be edited, the default state should be View mode (and Edit mode should be disabled).

- **Form captions:**
  - Avoid setting the form caption programmatically. Instead, consider setting the `TitleDataSource` property on the form design node to enable the framework to provide the caption dynamically.
  - If you can’t avoid setting the form caption programmatically, make sure that it’s short (no more than 30 characters). This guideline exists because a large font size is used for the form in some form types.
    - **Exceptions:** Custom Lookups, FactBoxes
  - Form captions should provide the user with the context of the “type” of entity. The font size and the position of the form caption will vary, depending on the type of form.
  - Don’t use the form caption to convey contextual information such as the parent record or other status information.

- All labels in the form are in sentence case. The framework guarantees consistency by putting some elements, such as Group labels, FactBox captions, Action Pane tab labels, and Button Group labels, in ALL CAPS. These strings should still be added in sentence case, but the framework will display them in all caps.

- All labels in the form are spelled correctly and use proper grammar.

- Avoid overriding the formatting alignment for extended data types (EDTs).

- When custom filters are used, no more than five should be specified. Instead, consider pre-populating the filter pane with fields.

- Occasionally, a decision must be made to determine whether a control should be temporarily disabled or hidden. The following information can help you make this determination:
A control should be temporarily disabled if the specific conditions must be met before the control can be enabled, and the user must first perform some action to meet those conditions.

A control should be hidden from the user if there is nothing that the user can do to enable or edit the control.

**Exception:** The control is used to convey status to the user.

- No UX guidelines are violated when security is applied for the various roles that have access to this form.

**Example:** For role A, field A is not required, but for role B, field A is required.

- No UX guidelines are violated when country/region codes are applied.

- Two fields can share a single label. Group the fields into a group, and set the **FrameType** property of the group to **GroupedFieldsLabel**.

![Financial Dimensions](image)

**Other form guidelines**

- Use a StaticText control instead of StringEdit for multi-line read-only text. StringEdit controls are semantically incorrect for informational text, because they can never be edited. Additionally, StringEdit controls typically have a border and different layout characteristics than StaticText controls, and these differences negatively affect the user experience.

- Controls that are always read-only but data-bound should be marked as **ViewEdit = View**.

- (Dialogs and Drop Dialogs only) The button that is chosen as the default button should be the safest, most secure response to the task that the user is performing. For example, the default button should correspond to the main instruction of a Dialog or Drop Dialog.

  - If safety and security aren’t factors, the button that is most likely to be clicked or that is most convenient for the user should be chosen as the default button.

  **Exception:** Don’t select a destructive response as the default button, unless there is an easy, obvious way to undo the command.

**Field state guidelines**

It’s important that the state of a field be set correctly. The state of the field communicates key information about what the user can do and how to do it. The following table outlines key guidelines about when to use the various field states. **Note:** In the past, field states weren’t used correctly. People interchangeably used **Enabled = No** and **ReadOnly = Yes**. Note the semantic differences between the two states:

- **Enabled = No** – The data is not valid.

- **ReadOnly = Yes** – The data is valid.
<table>
<thead>
<tr>
<th>STATE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| Enabled=No       | Disabled fields tell the user that the field isn't valid in the current state of the entity. Use the **Enabled** property to disable a field and prevent data input. A disabled field ([Enabled]=No) has these characteristics:  
  • The field is visually presented so that it looks unavailable. The text will be gray to indicate a “not valid” appearance.  
  • The field’s value is unimportant and won’t be sent to the server for processing.  
  • The field is skipped in the tab sequence.  
  
  A disabled field might become enabled by actions that the user takes in the form. If the user can never enable a field, consider hiding the field instead. |
| Readonly=Yes     | For read-only fields, the data is valid in the current context but can’t be edited. The field isn’t skipped in the tab sequence. If the value can never be edited, consider using **ViewEditMode=View** for the field. |
| ViewEditMode=View| Use this state when the value in a field can never be edited by the user and is for informational purposes only.                                 |
| Secured          | Fields are typically hidden if they are secured. In grids, each row can define different levels of security for each column. Therefore, for cells where the user has no access, a padlock appears in the field. This isn’t an application-controlled ability and occurs automatically. |
| Not available    | Similar to secured fields, data can appear in a grid where each row has a different set of columns. In this case, cells that aren’t applicable to a row display a “not” symbol. This isn’t an application-controlled ability and occurs automatically. |

**Mandatory field guidelines**

Mandatory fields are fields that the user must supply values for to guarantee database referential integrity and business logic integrity. By marking fields as mandatory, you provide an important indicator about what the user must do.

- All mandatory fields are marked.
- Mandatory fields should be set on metadata concepts in this order:
  1. Table
  2. Data source
  3. Form field
- Mandatory fields should be set programmatically, based on the state of the record. For example, a field that is required in order to post an entity should be marked as mandatory.
  - If the user receives a validation message about an empty field, but that field isn’t marked as mandatory, the user experience is negatively affected.

**FastTabs guidelines**

- The fields in groups should flow across the FastTab.
The content of the first FastTab should be fully visible without scrolling. FastTabs should never horizontally scroll when the fields are displayed.

The first FastTab should contain the most important fields for this entity (the fields that will be edited most often).

FastTabs should display summary information.

- **Exceptions:**
  - FastTabs that contain only a grid aren’t expected to display summary fields.
  - Dimensions FastTab pages can’t display summary fields, because this functionality isn’t currently supported.

- If the FastTab contains a grid, it should follow the Toolbar and List subpattern guidelines.

**Radio button guidelines**

- Follow all standard Microsoft guidelines for radio buttons. Specifically, observe these guidelines:
  - The radio button control is used to select one option from a set of mutually exclusive choices.
  - There are between two and seven choices. If there are more than seven choices, use a combo box instead.
  - If none of the options is a valid choice, there is another option to reflect this situation, such as None or Does not apply.
  - Selection of a radio button doesn’t perform commands, display other windows, such as a dialog box, or dynamically show/hide other controls that are related to the selection.
  - One radio button option is always selected by default. (However, see the exceptions at the MSDN guidelines link.)
  - The default option is the safest (that is, the option prevents loss of data or system access) and most secure and private option that is available. Alternatively, the default option is the most likely and convenient option.
  - Every radio button has a label.

- If subordinate controls are required for the selection of a radio button, follow these guidelines:
  - The subordinate controls are visible and appear to the right of or below the radio button.
  - Using the subordinate control selects the radio button.
  - The subordinate controls aren’t nested radio buttons that have other radio buttons or check boxes.

- Options are listed in a logical order:
  - From the most likely to be selected to the least likely to be selected
  - From the simplest operation to the most complex operation
  - From the least risk to the most risk

**Check box and toggle guidelines**

Toggle buttons are typically used instead of traditional check boxes.
Selection panel guidelines

- Occasionally, you'll require a discrete list of options that the user can select from, where each option presents the same interface. Don't use radio buttons to present the list of options. Instead, use a combo box control inside a Custom Filter Group.
- If the selection list is common to a set of items, the combo box should appear above all the items that are affected.

General grid guidelines

- Grids should be sorted on the first column and in ascending order, unless the scenario requires a different sort order:
  - The identifier (ID) field for documents (Sales Order ID, Purchase Order ID, and so on)
  - The name/description field for entities (Vendor name, Customer name, and so on)
  - Depending on the scenario, sorting can be on another column that makes business sense, such as the sequence number or date.
- **Editable grids:**
  - You must show and set the mandatory fields in the grid. It's not acceptable to alert the user to a missing mandatory field only when the record is saved.
  - All editable grids must have a New/Delete or Add/Remove button in a toolbar above the grid, or in the global Action Pane.
- Order the columns so that the most important columns are on the left. Amount columns are most usable for a user when they are placed in the rightmost position in the grid.
- Image columns can be used to convey state information for an entity or process, such as the workflow status.
- In general, the image column should be placed on the left side of the grid, and there should not be a column label. The meaning of the image should be indicated by the tooltip.
- Numeric columns should be right-aligned. All other columns should be left-aligned.
- If the form can be opened in View mode and can be used to create records in Edit Mode, the property of the

Follow all standard Microsoft guidelines for check boxes. Specifically, observe these guidelines:

- By default, use toggle buttons instead of check boxes in forms. The label must follow the Microsoft guidelines for check box labels.
  - **Exceptions:**
    - Use a check box when a large number of related options must be set in a group.
    - Use a check box inside Custom Filter Group subpatterns.
    - Groups where a check box is used on the frame to collect related fields. *This exception is currently under review to add clarity.*
  - Don't use check boxes/toggle buttons as a progress indicator.
  - Don't use check boxes to initiate a command. However, a message box can be shown to the user to refine or clarify the choice.
  - Write the label so that it describes the selected state of the check box. The meaning of the cleared state must be the unambiguous opposite of the selected state.
  - For a group of check boxes, use similar phrasing, and try to keep the length of all labels about the same.
  - Use positive phrasing. Don't phrase a label so that selection of the check box means that an action is not performed.

Occasionally, you'll require a discrete list of options that the user can select from, where each option presents the same interface. Don't use radio buttons to present the list of options. Instead, use a combo box control inside a Custom Filter Group.

If the selection list is common to a set of items, the combo box should appear above all the items that are affected.
root data source that is connected to the grid should be Insert If Empty = No. This setting will prevent situations where a user sees and selects a blank record when a read-only form is opened.

Entity status field guidelines

- The entity status must appear in the upper-right of the form, to the right of the title fields. The Details form patterns provide an optional Group where status fields can be defined.

Standard Action Pane guidelines

- The Standard Action Pane is at the top of the form and follows the standard Action Pane guidelines.
  - **Exceptions:** Dialog, Drop Dialog, Lookups
- For application-added actions, labels for similar actions should be used consistently across all Action Panes.
- If the name of the button doesn't adequately explain the action, a supplemental description should be shown in the tooltip.
- Actions on an Action Pane should apply to the whole entity. Never put actions that apply only to portions of the entity on the Action Pane. Instead, use local toolbars that are near the objects that will be acted on.
- There should be no more than 10 tabs on a Standard Action Pane.
  - There should be between one and eight actions per group.
  - The first tab is the home tab. It should have the same name as the entity and should be singular.
- **Activity tabs:**
  - Actions that are related to a specific activity should be grouped into an appropriately named Activity tab.
  - These tabs should be given the names of activities that the user will understand. The names should consist of action verbs.
  - Remove system-defined actions from Activity tabs.
  - **New** and **Maintain** groups group actions, such as **New**, **Delete**, **Edit**, and **Save**, that are related to these primary actions. The exception is **New** actions that are related to secondary types within the entity.
- **General tab:**
  - If there are common, infrequent actions that aren’t related to a specific activity, they should appear on the last tab, which should be named **General**.
  - Commands that appear on the **General** tab should not be repeated on other tabs.

Buttons on canvas guidelines

- Buttons in the content area of a form that are related to the form, not to a specific field, should be placed on the standard Action Pane or in a Toolbar.

Button image guidelines

- Buttons that require images should use symbols for their images.
- If both an image and text are shown on a button, the image must be to the left of the text. No other configurations are supported.
- **Standard Action Pane:**
  - Buttons that replace the system **New/Delete** buttons should use the New/Delete symbols.
  - Common actions should have **ButtonDisplay = Auto**, unless they have symbols that are used by other system buttons. In that case, they should be **TextOnly**. Consult UX if there are other common actions that you believe should have symbols assigned.
Other uncommon actions should be **TextOnly**.
Images aren't supported on Action Pane tabs.

- **Toolbars:**
  - **Add/Remove** (if applicable) should be the first buttons, should use the Add/Remove symbols, and should have both images and text (**ButtonDisplay** = **Auto**).
  - Common actions should be moved just to the right of the **Add/Remove** buttons. These actions can also have both images and text (**ButtonDisplay** = **Auto**). If no appropriate image exists, these can be **TextOnly**.
  - Subsequent uncommon actions should be **TextOnly**.

- **Inside MenuButtons:**
  - Images aren't supported on buttons in MenuButtons.

Appendix

**Frequently asked questions**

This section will have answers to frequently asked questions that are related to this guideline/pattern.

- **What types of fields aren't supported as summary fields on Fast Tabs?**
  - For performance reasons, we don't currently support reference groups and display methods. Additionally, unbound fields can't be used as summary fields.
This topic provides information about the Details Master form pattern. A details form is the primary method for entering data.

Usage

A details form is the primary method for entering data. These forms let the user view, edit, and act upon data. All content on these form types is structured into FastTabs that can be expanded and collapsed, so that multiple FastTabs can be open at the same time. The FastTabs can contain fields or a grid, and each FastTab can have a local toolbar. Two patterns are described in this document:

- **Detail Master** – This is the basic Detail Master pattern. This is the pattern that you should use by default.
- **Detail Master w/ Tabs** – You should use this pattern when an entity requires many FastTabs (more than 15) that can be grouped into categories.

In both cases, the grid view is structured the same.

Wireframe

**Details Master**

Details view

![Details Master wireframe](image)

**Grid view**

![Grid view](image)
Details Master with Standard Tabs

Details view

Grid view
Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- Added a List style grid to the left of the Details view content.
- Merged List Page and Details Master into a single form.
  - Improves performance when moving between a list and details.
  - Enables bulk editing in the initial list.
  - Allows for elimination of the list page preview pane.
- View/Edit, New, Delete, Save, Refresh, Attachments, and Export to Excel actions are all provided by the foundation and should not have explicit app buttons unless the foundation-provided button is removed.
- Master Details forms that previously used the TOC extension should now use the Master Details w/Standard Tabs pattern.

Model

Details Master (basic) – High-level structure

- Design
  - ActionPane (ActionPane)
  - SidePanel (Group)
    - QuickFilter
    - CustomFilters (Group) [Optional]
    - NavigationList (Grid, Style=List)
  - MainTab (Tab ShowTabs=No)
  - DetailsTabPage (TabPage)
    - TitleGroup (Group)
      - HeaderTitle (String)
      - EntityStatus (Group) [Optional]
Core components

1. Apply the DetailsMaster pattern on `Form.Design`.
2. Address BP Warnings:
b. Form must be referenced by at least one menu item.
c. `TabPage.Caption` isn't empty.

### Related patterns
- Details Transaction
- Simple List and Details

### Commonly used subpatterns
- Fields and Field Groups
- Toolbar and List
- Toolbar and Fields
- Nested Simple List and Details
- Custom Filter Group

### UX guidelines
The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in a browser, and walk through these steps. **Standard form guidelines:**

- Standard form guidelines have been consolidated into the Microsoft Dynamics AX General Form Guidelines document.

**Detail Master guidelines:**

- There should not be any duplicate `New` and `Delete` buttons.
- Should use FastTabs to group the fields instead of traditional tabs. The Details Master w/Standard Tabs pattern groups these related FastTabs into traditional tabs.
  - In its `default` state, the content of the first FastTab should be fully visible without scrolling.
  - FastTabs guidelines have been consolidated into the General Form Guidelines document.
- **ActionPane guidelines** have been consolidated into the General Form Guidelines document, in the ActionPane guidelines section.

**Page title area:**

- The following format should be used: "<ID> : <Description>"
- A link to the Details page should be provided in the Main Menu when the List page has been merged into the Details page.
- The page title should be in a plural form.
- **FactBox guidelines** have been consolidated into the FactBox Form Patterns document.

**Navigation list grid:**

- The list style grid should not have fields within a grid row that cause the row to span more than three lines.
  - Typically, just the ID and Description are sufficient.
  - There should be at least two fields.

**Grid view:**

- The grid has 2 to 15 fields. Typically, all mandatory fields are included, so that records can be created in the grid.
- A linked field lets the user open the details for the selected record.
- The Quick filter should default to the most likely field for a filter scenario.
- **Grid:**
  - The `ID` field should be the first column (if it's needed in the grid). Otherwise, the `Name` field should be the first column.
Additional grid guidelines have been consolidated into the General Form Guidelines document, in the Grid guidelines section.

Examples

Details Master (basic)
Form: CustTable

Details view (navigation list off)

Details view (navigation list on)

Grid view
Details Master with Standard Tabs

Form: HcmWorker

Details view (navigation list off)

Details view (navigation list on)
Appendix

Frequently asked questions
This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues
None.

AX 2012 content

AX 2012 links
- AX 2012 MSDN Details Forms

AX 2012 example
Details Master (basic)
Details Master with Standard Tabs
This topic provides information about the Details Transaction form pattern. Forms that use this pattern can have two details views that the user can switch between - a Header view and a Line view.

Usage

A details form with lines (Details Transaction form) consists of one form that can have two details views that the user can switch between. The Header view contains all fields that are related to or part of the header. The Line view contains the lines grid, line details, and a section that contains a collection of the most important header fields.

Wireframe

**Line view**

![Line view wireframe](image)

**Header view**

![Header view wireframe](image)
Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- A list style grid has been added to the left of the Details view content, which is shown in either Header view or Line view.
- List Page and Details Master have been merged into a single form. This change has the following benefits:
  - It improves performance when users move between the list and details.
  - It enables bulk editing in the initial list.
  - It allows for elimination of the list page preview pane.
- View/Edit, New, Delete, Save, Refresh, Attachments, and Export to Excel actions are all provided by the foundation and should not have explicit app buttons unless the foundation-provided button is removed.
Model

High-level structure

- Design
  - ActionPane (ActionPane)
  - SidePanel (Group)
    - QuickFilter
      - CustomFilters (Group) [Optional]
      - NavigationList (Grid, Style=List)
  - PanelTab (Tab ShowTabs=No)
    - DetailsPanel (TabPage)
      - TitleGroup (Group)
        - HeaderTitle (String)
        - EntityStatus (Group) [Optional]
          - StatusFields (1..N)
        - HeaderLinePanels (Tab ShowTabs=No)
          - LinePanel (TabPage PanelStyle=Line)
            - LineViewTab (Tab Style=FastTabs)
              - LineViewHeader (TabPage)
              - LineViewLines (TabPage)
              - LineViewLineDetails (TabPage)
                - LineDetailsTab (Tab Style=Standard)
                - LineDetailsTabPages (TabPages 1..N)
          - HeaderPanel (TabPage PanelStyle=Header)
            - HeaderViewTab (Tab Style=FastTabs)
            - HeaderViewTabPages (TabPages 1..N)
    - GridPanel (TabPage PanelStyle=Grid)
      - CustomFilterGroup (Group)
        - QuickFilter
        - OtherFilters ($Field) [0..N]
      - MainGrid (Grid)
      - MainGridDefaultAction (CommandButton)

Core components

1. Apply the DetailsTransaction pattern on Form.Design.
2. Address BP Warnings:
   b. The form must be referenced by at least one menu item.
   c. TabPage.Caption isn't empty.
d. TabPage.DataSource isn't empty.

**Related patterns**
- Details Master
- Simple List and Details

**Commonly used subpatterns**
- Fields and Field Groups
- Toolbar and List
- Toolbar and Fields
- Nested Simple List and Details
- Custom Filter Group

**UX guidelines**
The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps. **Standard form guidelines:**

- Standard form guidelines have been consolidated into the Microsoft Dynamics AX General Form Guidelines document.

**Detail Transaction guidelines:**
- There should not be any duplicate New and Delete buttons.
- ActionPane guidelines have been consolidated into the General Form Guidelines document, in the ActionPane guidelines section.
- In its default state, the content of the first FastTab should be fully visible without scrolling.
- FastTabs guidelines have been consolidated into the General Form Guidelines document.

**Page title area:**
- The following format should be used: `<ID> : <Description>
- A link to the Details page should be provided on the Main Menu after the List page has been merged into the Details page.
- The page title should be in a plural form.

**Navigation list grid:**
- The list style grid should not have fields within a grid row that spans more than three lines.
  - Typically, just the ID and Description are sufficient.
  - There should be at least two fields.
- The last field should be the total of the transaction.

**Grid view:**
- The grid has 2 to 15 fields. Typically, all mandatory fields are included, so that records can be created in the grid.
- A linked field lets the user open the details for the selected record.
- By default, the Quick Filter should use the most likely field for a filter scenario.
- Focus should be in the Quick Filter when the list page is opened.

**Grid:**
- The ID field should be the first column, followed by the master entity ID and Name fields.
- Additional grid guidelines have been consolidated into the General Form Guidelines document, in the Grid guidelines section.
- FactBox guidelines have been consolidated into the FactBox Form Patterns document.
Example

Form: SalesTable

Line view

Header view

Grid view
Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

- **Why is the Header view compulsory?**
  
  The Header view is compulsory for the Details Transaction pattern. Initially, the Header view might not have more than the Line view header summary information. However, over time, it will be extended by application teams, internationalization teams, partners, and customers. It's important that the Header view be available for future modifications. In addition, a consistent and dependable form structure has benefits for usability and upgrade reasons.

- **Why are the Header/Lines buttons no longer to the right of the record title at the top of the page?**
  
  The Header/Lines buttons in the header portion of the page were radio buttons restyled to mimic tabs. For improved accessibility on these pages, the **Removal of header/lines proxy buttons** feature removes these radio buttons and instead surfaces the native tab controls that are under the record title to allow you to switch between header lines. Before enabling this feature, you should evaluate the impact of this feature on test assets and task recordings.

**NOTE**

This feature is included in the platform updates for version 10.0.23 of Finance and Operations apps.

Open issues

None currently.

AX 2012 content

AX 2012 links

- MSDN Details Form with Lines User Experience Guidelines [AX 2012]
- MSDN Details Form [AX 2012]

AX 2012 example

Line view
<table>
<thead>
<tr>
<th>ID</th>
<th>Customer account</th>
<th>Invoice account</th>
<th>Order type</th>
<th>Continuity order</th>
<th>Status</th>
<th>Currency</th>
<th>Confirmed receipt date</th>
<th>Confirmed</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD-101246</td>
<td>2003</td>
<td>2003</td>
<td>Sales order</td>
<td>Open order</td>
<td>USD</td>
<td>Open order USD</td>
<td>9/13</td>
<td>9/13</td>
<td>101246</td>
</tr>
<tr>
<td>SD-101247</td>
<td>2121</td>
<td>2121</td>
<td>Sales order</td>
<td>Open order</td>
<td>USD</td>
<td>Open order USD</td>
<td>8/22</td>
<td>8/22</td>
<td>101247</td>
</tr>
<tr>
<td>SD-101248</td>
<td>1104</td>
<td>1104</td>
<td>Sales order</td>
<td>Open order</td>
<td>USD</td>
<td>Open order USD</td>
<td>7/11</td>
<td>7/11</td>
<td>101248</td>
</tr>
<tr>
<td>SD-101249</td>
<td>1101</td>
<td>1101</td>
<td>Sales order</td>
<td>Open order</td>
<td>USD</td>
<td>Open order USD</td>
<td>7/11</td>
<td>7/11</td>
<td>101249</td>
</tr>
<tr>
<td>SD-101250</td>
<td>1102</td>
<td>1102</td>
<td>Sales order</td>
<td>Open order</td>
<td>USD</td>
<td>Open order USD</td>
<td>7/11</td>
<td>7/11</td>
<td>101250</td>
</tr>
<tr>
<td>SD-101251</td>
<td>1103</td>
<td>1103</td>
<td>Sales order</td>
<td>Open order</td>
<td>USD</td>
<td>Open order USD</td>
<td>7/11</td>
<td>7/11</td>
<td>101251</td>
</tr>
<tr>
<td>SD-101252</td>
<td>1104</td>
<td>1104</td>
<td>Sales order</td>
<td>Open order</td>
<td>USD</td>
<td>Open order USD</td>
<td>7/11</td>
<td>7/11</td>
<td>101252</td>
</tr>
<tr>
<td>SD-101253</td>
<td>1101</td>
<td>1101</td>
<td>Sales order</td>
<td>Open order</td>
<td>USD</td>
<td>Open order USD</td>
<td>7/11</td>
<td>7/11</td>
<td>101253</td>
</tr>
<tr>
<td>SD-101254</td>
<td>1202</td>
<td>1202</td>
<td>Sales order</td>
<td>Open order</td>
<td>USD</td>
<td>Open order USD</td>
<td>7/11</td>
<td>7/11</td>
<td>101254</td>
</tr>
<tr>
<td>SD-101255</td>
<td>1203</td>
<td>1203</td>
<td>Sales order</td>
<td>Open order</td>
<td>USD</td>
<td>Open order USD</td>
<td>7/11</td>
<td>7/11</td>
<td>101255</td>
</tr>
<tr>
<td>SD-101256</td>
<td>1301</td>
<td>1301</td>
<td>Sales order</td>
<td>Open order</td>
<td>USD</td>
<td>Open order USD</td>
<td>7/11</td>
<td>7/11</td>
<td>101256</td>
</tr>
<tr>
<td>SD-101257</td>
<td>1302</td>
<td>1302</td>
<td>Sales order</td>
<td>Open order</td>
<td>USD</td>
<td>Open order USD</td>
<td>7/11</td>
<td>7/11</td>
<td>101257</td>
</tr>
<tr>
<td>SD-101258</td>
<td>1304</td>
<td>1304</td>
<td>Sales order</td>
<td>Open order</td>
<td>USD</td>
<td>Open order USD</td>
<td>7/11</td>
<td>7/11</td>
<td>101258</td>
</tr>
<tr>
<td>SD-101259</td>
<td>2001</td>
<td>2001</td>
<td>Sales order</td>
<td>Open order</td>
<td>USD</td>
<td>Open order USD</td>
<td>7/11</td>
<td>7/11</td>
<td>101259</td>
</tr>
<tr>
<td>SD-101260</td>
<td>2002</td>
<td>2002</td>
<td>Sales order</td>
<td>Open order</td>
<td>USD</td>
<td>Open order USD</td>
<td>7/11</td>
<td>7/11</td>
<td>101260</td>
</tr>
<tr>
<td>SD-101261</td>
<td>2003</td>
<td>2003</td>
<td>Sales order</td>
<td>Open order</td>
<td>USD</td>
<td>Open order USD</td>
<td>7/11</td>
<td>7/11</td>
<td>101261</td>
</tr>
<tr>
<td>SD-101262</td>
<td>2004</td>
<td>2004</td>
<td>Sales order</td>
<td>Open order</td>
<td>USD</td>
<td>Open order USD</td>
<td>7/11</td>
<td>7/11</td>
<td>101262</td>
</tr>
<tr>
<td>SD-101263</td>
<td>2011</td>
<td>2011</td>
<td>Sales order</td>
<td>Open order</td>
<td>USD</td>
<td>Open order USD</td>
<td>7/11</td>
<td>7/11</td>
<td>101263</td>
</tr>
<tr>
<td>SD-101264</td>
<td>2012</td>
<td>2012</td>
<td>Sales order</td>
<td>Open order</td>
<td>USD</td>
<td>Open order USD</td>
<td>7/11</td>
<td>7/11</td>
<td>101264</td>
</tr>
</tbody>
</table>
This topic provides information about the Form Part Section List form patterns. These workspace-specific patterns have been developed to show filtered lists inside workspaces.

Usage

The Form Part Section List form patterns are workspace-specific patterns that are used to show filtered lists. The tabbed section of the workspace contains a set of vertical tabs. Each tab contains a Form Part Control that points to a form that contains one of the Form Part Section List patterns. Two patterns are described in this article:

- **Form Part Section List** – This is the default Section pattern. It allows for a single list of data, together with an optional header group that contains filters and/or actions. Most content areas in the tabbed section of a workspace will use this pattern.

- **Form Part Section List - Double** – This variant enables a second list of data to appear to the right of the primary list. By default, the secondary list is hidden. To show it, the user clicks a button on the Toolbar above the primary list.

Wireframe

**Form Part Section List**

<table>
<thead>
<tr>
<th>Form part</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HeaderGroup (Optional)</strong></td>
</tr>
<tr>
<td>Grid</td>
</tr>
<tr>
<td><strong>SeeMoreButton (Optional)</strong></td>
</tr>
</tbody>
</table>

**Form Part Section List - Double**
Pattern changes for Finance and Operations

These patterns did not exist for Microsoft Dynamics AX 2012.

Model

Form Part Section List: High-level structure

- Design | Container
  - Header (Group) [Optional] – This must use one of the Filters and Toolbar subpatterns.
  - Grid
  - GridDefaultAction (Button) [Optional]
  - SeeMoreButton (Button) [Optional]

Form Part Section List - Double: High-level structure

- Design | Container
  - PrimaryGroup (Group)
    - Header (Group) [Optional] – This must use one of the Filters and Toolbar subpatterns.
    - Grid
    - GridDefaultAction (Button) [Optional]
    - SeeMoreButton (Button) [Optional]
  - SecondaryGroup (Group)
    - Header (Group) [Optional] – This must use one of the Filters and Toolbar subpatterns.
    - Grid
    - GridDefaultAction (Button) [Optional]
    - SeeMoreButton (Button) [Optional]

Core components

1. Apply the appropriate Form Part Section List pattern on Form Design.
2. In the backing Operational workspace form, set the Form Part control on the corresponding vertical tab to point to a menu item that points to this form.

Related container patterns

- Section Tabbed List
UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

- **General form guidelines**
  - Standard form guidelines have been consolidated into the General Form Guidelines document.

- **Pattern-specific guidelines**
  - If a backing form exists, and especially if not all the records are shown in the list, a See more button should appear at the bottom of the list, so that the user can see the full list.
  - Up to two important filters exist above the list.
  - Up to three frequently used actions exist above the list.

- **Grid**
  - Lists are filtered down to an interesting, relatively small set of data.
  - List grids have no more than three lines of data per row.
  - Card grids show no more than four fields (not including an image).
  - Tabular grids show no more than eight fields.

- **Form Part Section List - Double guidelines**
  - If both lists have actions and/or filters, both list must use the same Filters and Toolbar subpattern (either the Stacked variant or the Inline variant).

Examples

**Form Part Section List**

Form: PurchOrderProcessReceiptsWorkspace > PurchOrdersWithDelayedReceiptsPart (All workspaces > Purchase order receipt and follow-up)
Form Part Section List - Double

Form: BudgetTrackingWorkspace > BudgetTransactionPart (All workspaces > Ledger budgets and forecasts)
Appendix

**Frequently asked questions**

This section will have answers to frequently asked questions that are related to this guideline/pattern.

**Open issues**

None
This article provides information about the List Page form pattern. A list page presents a set of data on a UI that is optimized for browsing records, so that you can find and work with a specific record.

Usage

A list page presents a set of data on a user interface that is optimized so that you can browse records, find the right record, and then take an action upon that record. The list page lets the user search, filter, and sort the data. FactBoxes on the right side of the grid show related data for the active record. Actions that are relevant to the record are located on the ActionPane at the top of the page. The use of this pattern is now discouraged when there is a 1:1 correspondence between the List Page and Details page. Current guidance is to use this pattern only in other situations, such as when list pages have no backing details pages or have multiple backing details page (for example, when project quotations and sales quotations are shown together in the same List Page).

Wireframe

![List Page wireframe](image)

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- FormTemplate/InteractionClass is now optional when you build new pages.
- List Page and Details Master/Details Transaction are merged into a single form when there is a 1:1 correspondence between the List Page and Details Page.
  - Improves performance when the user moves between the list and details.
  - Allows for bulk editing in the initial list.
- The Preview pane has been eliminated.

Model
High-level structure

- Design
  - ActionPane (ActionPane)
  - Custom Filter (Group)
    - Quick Filter (Quick Filter)
    - OtherFilters ($Field) [0..N]
  - Grid (Grid)

Core components

1. Apply the ListPage pattern on Form Design.
2. Address BP Warnings:
   b. The form must be referenced by at least one menu item.
   c. TabPage.Caption isn't empty.
   d. TabPage.DataSource isn't empty.
   e. The primary data source has AllowEdit = No, AllowCreate = No, and AllowDelete = Yes.
   f. Grid.DefaultAction references the button that opens the child form.
   g. Grid.DefaultLabelAction references a label to show in the grid context menu.

Related patterns

- Details Master
- Details Transaction
- Simple List

Commonly used subpatterns

- Custom Filter Group

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps. Standard form guidelines:

- Standard form guidelines have been consolidated into the General Form Guidelines document.

List Page guidelines:

- Have fewer than 15 fields in the grid.
- The first textual/data column should be displayed as a link that goes to the appropriate details form. To do this, make sure that the grid has a default action to enable the hyperlink for the first column.
- A Quick Filter should appear above the list. By default, the QuickFilter should use the most likely field for a filter scenario.
- There should not be any duplicate New and Delete buttons.
- A link to the List page should be provided in the Main Menu.
- Focus should be in the Quick Filter when the list page is opened.

Page title area:
  - The page title should be in a plural form.
  - For primary list pages, the title should be the name of the entity.
  - For secondary list pages, the title should reflect an activity or status.

Grid:
- For transactional entities, the **ID** field should be the first column, followed by the master entity **ID** and **Name** fields.
- For master entities, the **Name** field should be the first column, followed by the **ID** field.

- **ActionPane** guidelines have been consolidated into the Dynamics AX General Form Guidelines document in the ActionPane guidelines section.
- **FactBox** guidelines have been consolidated into the FactBox Form Patterns document.

### Examples

**Form: SalesTableListPage**

![SalesTableListPage example](image)

- **Appendix**

  **Frequently asked questions**

  This section will have answers to frequently asked questions that are related to this guideline/pattern.

  - **What do I do with the Preview pane when I migrate the form?**

    You can do one of the following:

    - Remove the **Preview** pane altogether if it no longer makes sense.
    - Remove the large header, and leave the **Preview** pane as is.
    - Split the **Preview** pane into multiple logical FactBoxes if the current one is too tall.
      - For a transaction preview, the lines would go into their own FactBox and should be limited to five lines.
      - For a transaction preview, rework the lines into a FactBox card pattern, where the lines are summarized into a count and a lines grid is shown in an enhanced preview when the user hovers over the count value.

  **Open issues**

  - **How to handle secondary list pages**
    - Stay in navigation (no app changes needed).
    - Create role-tailored views (after future framework support is added).

  **AX 2012 content**

  **AX 2012 links**

  - [MSDN AX 2012 List Page User Experience Guidelines](#)
This article describes the Simple Details form pattern. This pattern is used when only a simple set of fields must be presented to the user.

Usage

The Simple Details pattern is used when only a simple set of fields must be presented to the user. Examples include the display of totals and customer balances. Typically, view mode is used for the Simple Details pattern. However, in cases where the form provides editable information, the edit mode should be synced to the parent form. Four patterns are described in this document:

- **Simple Details w/Toolbar and Fields** – This is the basic Simple Details pattern, in which several fields are displayed in the form. The fields can optionally appear inside Groups.
- **Simple Details w/Fast Tabs** – This is the Simple Details pattern that should be used when fields are organized into FastTabs.
- **Simple Details w/Standard Tabs** – This is the Simple Details pattern that should be used when fields are organized into traditional tabs.
- **Simple Details w/Panorama** – This is the Simple Details pattern that should be used when information is intended to be displayed in a panorama format.

Wireframe

![Form Caption](image)

Pattern changes

There are no planned changes for the use of this pattern in the current version of Microsoft Dynamics AX.

Model
Simple Details w/Toolbar and Fields – High-level structure
- Design
  - ActionPane (ActionPane)
  - Body (Group) – Note: A field subpattern is used.

Simple Details w/FastTabs – High-level structure
- Design
  - ActionPane (ActionPane)
  - *HeaderGroup (Group) [Optional]*
  - Body (Tab, Style=FastTabs)
    - BodyTabPages (TabPage repeats 1..N)
  - *FooterGroup (Group) [Optional]*

Simple Details w/Standard Tabs – High-level structure
- Design
  - ActionPane (ActionPane)
  - *HeaderGroup (Group) [Optional]*
  - Body (Tab, Style=Tabs)
    - BodyTabPages (TabPage repeats 1..N)
  - *FooterGroup (Group) [Optional]*

Simple Details w/Panorama – High-level structure
- Design
  - ActionPane (ActionPane)
  - Body (Tab, Style=Panorama)
    - BodyTabPages (TabPage repeats 1..N)
  - *FooterGroup (Group) [Optional]*

Core components
1. Apply the SimpleDetails pattern on Form.Design.
2. Address BP Warnings:
   b. The form must be referenced by at least one menu item.
   c. TabPage.Caption isn't empty.
   d. MainMenu must not contain menu items that reference a SimpleDetails form.

Commonly used subpatterns
- Fields and Field Groups
- Toolbar and Fields
- Tabular Fields
- Toolbar and List

UX guidelines
The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This
checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps. **Standard form guidelines:**

- Standard form guidelines have been consolidated into the Dynamics AX General Form Guidelines document.

**Simple Details guidelines:**

- The form page should display a Form Caption that accurately describes the entity.
  - The Form Caption should be in a singular form.

**Examples**

**Simple Details w/Toolbar and Fields**

*Form: AgreementLine*

![Form](image)

**Simple Details w/FastTabs**

*Form: PlanActivityServiceDetails*
Simple Details w/Standard Tabs

Form: HcmEmploymentDateManager (Click Human Resources > Common > Workers > Workers, click General > Versions > Employment History, and then click Date Manager.)

Simple Details w/Panorama

Form: PdsMRCEventTracker
Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

- Investigate whether Simple Details forms that show a small amount of related content should have a different presentation than a full-page form.
This topic provides information about the Simple List and Details form pattern. This pattern is used to maintain data for entities of medium complexity.

Usage

The Simple List and Details (SL+D) pattern is used to maintain data for entities of medium complexity. Entities of medium complexity are those entities that have six or more fields. The Simple List pattern should be used for simple entities that have fewer than six fields. There are some exceptions where entities that have up to 15 fields are still considered simple entities. The Simple List and Details pattern is prescribed when these conditions are met:

- The underlying data has more than six fields.
- There are between zero and five child data collections.

Three patterns are described in this document:

- **Simple List and Details – List Grid** – This is the basic SL+D pattern. This is the pattern that should be used by default.
- **Simple List and Details – Tabular Grid** – This is the SL+D pattern that should be used if the number of fields in the “simple list” part of the form is larger than expected (see the “Pattern changes” section later in this article).
- **Simple List and Details – Tree** – This is the SL+D pattern that should be used if the “simple list” part of the form is actually a tree.

Wireframe

![Wireframe Image](image)

Pattern changes
Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The top ActionPane strip control has been converted to a standard ActionPane.
- **New**, **Delete**, and **Edit** buttons are provided by the framework.
- View mode is used by default.
- A Quick Filter control has been added above the “list” part of the form.
- Whenever possible, use the list-style grid for the “list” part of the form. A tabular grid is an acceptable alternative in some situations, such as when these conditions are met:
  - Multiple fields of the same type (for example, three date fields) would not be distinguishable in the list-style grid.
  - The user’s task is to compare a sequence of dates/numbers across rows in the list (for example, date effective dates or route step numbers).
  - The number of fields in the grid is larger than expected (if each row takes up more than three lines in the list-style grid).
- Fields in the header group are arranged horizontally instead of vertically.
- FactBoxes are allowed.
- The form structure has been simplified (the BodyGroup container has been removed).

**Model**

**High-level structure**

- Design
  - ActionPane
  - NavigationList (Group)
    - Quick Filter
    - **CustomFilterGroup (Group) [Optional]**
    - ListStyleGrid (Grid) | Tree | TabularGrid (Grid)
    - **VerticalSplitter (Group) [only allowed for Tree or TabularGrid variants]**
  - DetailsHeader (Group)
  - DetailsTab (Tab)

**Core components**

1. Apply one of the SimpleListDetails patterns on **Form.Design**.
2. Resolve required BP checks:
   a. Set **Design.Caption** the same as the label that is used on the **Name** property of the table.
   b. Set **Design.Datasource** the same as **Grid.Datasource**.
   c. Set the primary data source to **InsertIfEmpty = No**.
   d. Set the primary **ActionPane.DataSource** the same as **Grid.Datasource**.
   e. Set **Grid.Datasource** to the primary data source.

**Commonly used subpatterns**

- Fields and Field Groups
- Toolbar and List
- Toolbar and Fields
- Nested Simple List and Details

**UX guidelines**
The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

**Standard form guidelines:**

- Standard form guidelines have been consolidated into the Microsoft Dynamics AX General Form Guidelines document.

**Simple list & detail guidelines:**

- The page should display a Form Caption that accurately describes the entity.
  - The Form Caption should be in plural form.
- There should not be duplicate **New** or **Delete** buttons.
- By default, the Quick Filter should use the name or description column.
- Guidelines for custom filters have been consolidated into the Custom Filter Group subpattern document.
  - There should be no more than two custom filter fields in a SL&D form.
- There should be a tabular grid, a list-style grid, or a tree control on the left edge of the form.
  - List-style grids should display no more than three rows (lines) for each record in the List-style grid. Typically, just the ID and Description are sufficient.
  - Between two and five fields should be used for the list on the left.
  - A tabular grid can be used in some unique situations but isn’t generally recommended.
    - If a tabular grid is used, it should **not** be editable.
    - When there is no data, the grid or tree control should not automatically add a new record.
- A **Details** section should be displayed on the right of the form:
  - The list fields (whether they are from a list, tabular grid, or tree) should be the first fields in Header Group. They should appear in the same order that they appear in the grid or tree, so that the user can edit and see the labels of the fields.
- Simple List and Detail forms must **not** have these elements:
  - Standard tabs to group fields

**Examples**

**Simple List and Details – List Grid**

Form: **PaymTerm**
Simple List and Details – Tabular Grid

Form: ExchangeRate

Simple List and Details – Tree

Form: CaseCategorySetup
Appendix

Frequently asked questions
This section will have answers to frequently asked questions that are related to this guideline/pattern.

- **When do I use icons on actions in the toolbars?**
  - See the Button Image Guidelines in the General Form Guidelines document.

Open issues

- **How can a developer move between the ListStyleGrid and the TabularGrid patterns?**
  - Currently, developers must manually move between the patterns.

- **Are we going to allow modeling without FastTabs in the details body?**
  - Although we require FastTabs in the details body, we plan to eventually hide the FastTab header if only one FastTab is visible.

- **How do we allow for exceptions to the FastTab rule for legacy situations such as the Interest form?**
  - Whenever possible, refactor the form to fit the SL&D pattern (as the Interest form has done).
  - Otherwise, use custom containers.

- **How do we prevent hyperlinks on fields in the UI?**
  - For some fields, you can set IgnoreEDTRelation=Yes to prevent hyperlinks in the UI. Regardless (as of Platform update 17), you can set EnableFormRef=No on an input control to disable a hyperlink.

AX 2012 content
This article provides information about the Simple List form pattern. This pattern is used to maintain data for simple entities.

Usage

The Simple List pattern is used to maintain data for simple entities. Simple entities are entities that have six or fewer fields and no parent/child relationships. There are some exceptions where entities that have up to 15 fields are still considered simple entities.

Wireframe

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The top ActionPane strip control has been converted to a standard ActionPane.
- **New**, **Delete**, and **Edit** buttons are provided by the framework.
- View mode is used by default.
- A Quick Filter has been added above the grid.
- When the form is used as a dependent form, the parent form record context is automatically shown above the form caption.
  - The page title group for dependent form usage was removed, because it will be provided by the framework.
- The pattern allows for multiple selections in the grid.

Model
High-level structure

- Design
  - ActionPane (ActionPane)
  - Custom Filter (Group)
    - Quick Filter (Quick Filter)
    - OtherFilters ($Field) [0..N]
  - TabularGrid (Grid)
  - Footer (Group) [Optional]

Core components

1. Apply the SimpleList pattern on Form.Design.
2. Address BP Warnings:
   b. Design.DataSource isn't empty.
   c. Grid.Datasource must be set.
   d. The form must be referenced by at least one menu item.
   e. Design.Datasource is set the same as Grid.Datasource.
   f. The primary key field of the primary data source's table has IgnoreEDTRelation = Yes.
   g. The grid must not contain more than 15 fields.

Commonly used subpatterns

- Custom Filter Group

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

Standard form guidelines:

- Standard form guidelines have been consolidated into the Microsoft Dynamics AX General Form Guidelines document.

Simple list guidelines:

- By default, the Quick Filter should use the name or description column.
- The list can display up to 15 columns.
  
  Note: This guideline has been relaxed from AX 2012.
- There should not be any duplicate New or Delete buttons.
- The page title should be in a plural form.
- When there is no data, the grid should not automatically add a new record.

Examples

Form: CustGroup
Note: We plan to extend the grid lines to the right and bottom edges in a future client deliverable.

Appendix

Frequently asked questions
This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues
None at this time.

AX 2012 content
This article provides information about the Table of Contents form pattern. This pattern should be used when two or more logically related forms are required for setup configuration.

**Usage**

The Table of Contents pattern should be used when two or more logically related forms are required for setup configuration. The vertical arrangement of tabs implies the order of completion. This form pattern is also used for collections of unrelated items, such as tab pages that have a different root entity per tab. This form pattern contains a collection of smaller content regions, each of which follows a container subpattern such as Toolbar and List, Nested Simple List and Details, or Fields and Field Groups.

**Wireframe**

![Wireframe Diagram]

**Pattern changes**

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The Content Body child container uses dynamic columns for a responsive layout.
- An optional secondary instruction has been added under the Title Group.

**Model**

**High-level structure**

- Design
  - Tab (Style=VerticalTabs)
    - TabPage [repeats 1..N times]
Core components
- Apply the TableOfContents pattern on Form.Design.
- Address BP Warnings:
  - Design.Caption isn't empty.
  - The form must be referenced by at least one menu item.
  - TabPage.Caption isn't empty.
  - TabPage.DataSource isn't empty.
  - StaticText.Text isn't empty.

Commonly used subpatterns
Each BodyGroup will use one of the following container patterns for the content in the Table of Contents section:
- Fields and Field Groups
- Toolbar and List
- Toolbar and Fields
- Nested Simple List and Details
- Tabular Fields
- List Panel

UX guidelines
The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

Standard form guidelines:
- Standard form guidelines have been consolidated into the Microsoft Dynamics AX General Form Guidelines document.

Table of contents guidelines:
- The supplemental instruction, if it’s shown, is composed of a complete, concise sentence in sentence case and has end punctuation.
- TOC tabs should appear in the same sequence that is typically used to enter information.
- The first tab in the list should be highlighted when the form is opened, unless the form is opened in the context of a specific task from another form.
- The content area for the TOC content should primarily be one of three patterns: Simple List, Simple List and Details, or Simple Details.
  - Simple List content should follow the subpattern guidelines.
  - Simple List and Details content should follow the Nested Simple List and Details subpattern guidelines.
  - Simple Details content should follow the Toolbar and Fields subpattern guidelines.
  - FastTabs should follow the FastTab guidelines in the Dynamics AX General Form Guidelines document.
  - Actions appearing on a Toolbar on a tab page.
- A TOC form should not have the following:
- Application actions on a standard ActionPane. (It should have only framework actions.)
- FactBoxes.
- Standard tabs on a TOC tab page.

Examples

Form: CustParameters

Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

- What do I do with 'Global' buttons?
  - There have been several cases where a button is required in order to initialize data or sync information between services. Because we allow only system buttons on the standard Action Pane in this pattern, we recommend that these buttons go in one of two places:
    - On the tab page that the action is most closely related to.
    - If a place doesn’t exist, on a toolbar on the first tab page of the pattern.

Open issues
- None

AX 2012 content
### Accounts receivable parameters (2 - ceu)

**Set up requirements for sales order approval and customer information**

**Customer**
- **Mandatory tax group:**
- **Tax exempt number requirement:** None
- **Minimum reimbursement:** 0.00
- **One-time customer account:** 8011

**Sales**
- **Default values**
  - **Order type:** Sales order
  - **Period of validity:** DEF
  - **Sales order pool:**
  - **Reservation:** Manual
  - **Sales origin:**
  - **Sales origin from Enterprise Portal:**

**Setup**
- **Prompt for customer information:**
- **Prompt quantity field value when posting documents:**
- **Mark order as voided:**
- **Use billing classifications:**
- **Sales quotations**
  - **Days campaign expires:** 30

**Specify whether sales tax group for customer should be mandatory.**

### Benefit elements (1)

**Define benefit plans.** Benefit plans are a specific benefit that a provider is contracted to offer.

- **Plan:** 401(k)
- **Description:** Company 401(k) plan
- **Type:** Investment
- **Payroll impact:** Deduction and contribution

**Tax rule**
- **Plan:** United States
  - **401(k)**
  - **Custom**
    - **Custom pretax method:** By type
    - **Exempt from the following:**
      - Federal income tax:
      - State income tax:
      - Disability insurance:
      - Unemployment:
      - Medicare:

**Retirement plans:** 401(k)/Roth 401(k)
**Payroll details:** Year
**Accounting**
**Reporting**

The type of benefit, such as a medical or parking benefit.
This article provides information about the Task Double form pattern. This pattern was previously used to present a parent and child entity in the same form.

Usage

This type of form has previously been used when you wanted to present parent/child entities in the same form. This isn't a recommended pattern for new forms. No new forms should be created that use this pattern. This pattern will provide structure and stability for legacy forms, and will also provide a migration path to more modern form patterns.

Wireframe

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The form opens in view mode.
- The top ActionPane strip control has been converted to a standard ActionPane.
- The Overview label on the parent tab has been changed to List.
- The contents of the tab container use dynamic columns for a responsive layout.
- The label for the child tab's list should be `<x> list`, where `<x>` is replaced by an appropriate string, based on the entity. For example, if the child entity is usually called Charges, the label for the tab should be Charges list.
  - Exception: If the child entity is "lines" of some sort, the word "list" should not be added to the end.

Model
High-level structure

- Design
  - ActionPane (Action Pane)
  - CustomFilter (Group) [Optional]
  - ParentTab (Tab)
    - ParentList (TabPage) – **Note:** The Toolbar and List subpattern is used.
    - General (TabPage repeats 0..N)
  - ParentFooterGroup (Group) [Optional]
  - HSplitter (Group)
  - ChildToolbar (ActionPane) [Optional]
  - ChildTab (Tab)
    - ChildList (TabPage) – **Note:** The Toolbar and List subpattern is used.
    - General (TabPage, repeats 0..N)
    - ChildFooterGroup (Group) [Optional]

Core components

1. Apply the Task Double pattern on `Form.Design`.
2. Address BP Warnings:
   b. The form must be referenced by at least one menu item.
   c. `TabPage.Caption` isn’t empty.
   d. `TabPage.DataSource` isn’t empty.
   e. `StaticText.Text` isn’t empty.

Related patterns

- Task Single

Commonly used subpatterns

- Custom Filter Group
- Fields and Field Groups
- Toolbar and List
- Toolbar and Fields

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

Standard form guidelines:

- Standard form guidelines have been consolidated into the Microsoft Dynamics AX [General Form Guidelines](#) document.

Task Double guidelines:

- The Overview tab is the first tab and is active when the form is opened.
- The first tab on a child tab control should be called **Lines list** or an appropriate variation.
Selection in the parent grid will update content in the child grid.

Example

Form: HRMAbsenceTableHistory

Appendix

Frequently asked questions
This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues
- None

AX 2012 content
### Absence History

#### Absence Registration

<table>
<thead>
<tr>
<th>Absence registration</th>
<th>Worker</th>
<th>Start date</th>
<th>End date</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>000001.128</td>
<td>Charlie Carson</td>
<td>1/1/2007</td>
<td>1/31/2007</td>
<td>Approved</td>
</tr>
<tr>
<td>000004.128</td>
<td>Charlie Carson</td>
<td>4/1/2007</td>
<td>4/30/2007</td>
<td>Approved</td>
</tr>
<tr>
<td>000007.128</td>
<td>Charlie Carson</td>
<td>7/1/2007</td>
<td>7/31/2007</td>
<td>Approved</td>
</tr>
<tr>
<td>000008.128</td>
<td>Charlie Carson</td>
<td>8/1/2007</td>
<td>8/31/2007</td>
<td>Approved</td>
</tr>
<tr>
<td>000009.128</td>
<td>Charlie Carson</td>
<td>9/1/2007</td>
<td>9/30/2007</td>
<td>Approved</td>
</tr>
<tr>
<td>000010.128</td>
<td>Charlie Carson</td>
<td>10/1/2007</td>
<td>10/31/2007</td>
<td>Approved</td>
</tr>
<tr>
<td>000011.128</td>
<td>Charlie Carson</td>
<td>11/1/2007</td>
<td>11/30/2007</td>
<td>Approved</td>
</tr>
<tr>
<td>000012.128</td>
<td>Charlie Carson</td>
<td>12/1/2007</td>
<td>12/31/2007</td>
<td>Approved</td>
</tr>
<tr>
<td>000013.128</td>
<td>Charlie Carson</td>
<td>1/1/2008</td>
<td>1/31/2008</td>
<td>Created</td>
</tr>
<tr>
<td>000014.128</td>
<td>Charlie Carson</td>
<td>2/1/2008</td>
<td>2/29/2008</td>
<td>Created</td>
</tr>
</tbody>
</table>

#### Absence Code

<table>
<thead>
<tr>
<th>Date</th>
<th>Absence code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8/2...</td>
<td>Vacation</td>
<td>Vacation</td>
</tr>
<tr>
<td>8/2...</td>
<td>Vacation</td>
<td>Vacation</td>
</tr>
<tr>
<td>8/2...</td>
<td>Vacation</td>
<td>Vacation</td>
</tr>
<tr>
<td>8/2...</td>
<td>Vacation</td>
<td>Vacation</td>
</tr>
<tr>
<td>8/2...</td>
<td>Vacation</td>
<td>Vacation</td>
</tr>
</tbody>
</table>

The name of the worker: Charlie Carson
This article provides information about the Task Single form pattern. This pattern was previously used to present data that users would perceive as originating from a single data source that had multiple records.

Usage

This type of form was used when you wanted to present data that users will perceive as originating from a single data source with multiple records. This isn't a recommended pattern for new forms. No new forms should be created that use this pattern. This pattern will provide structure and stability for legacy forms, and will also provide a migration path to more modern form patterns.

Wireframe

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The form opens in view mode.
- Commands have been moved to the standard ActionPane from a Toolbar (ActionPane strips).
- The Overview label on the first tab has been changed to List.
- The content of the tab container uses dynamic columns for a responsive layout.

Model

High-level structure

- Design
  - ActionPane (Action Pane)
**Core components**

1. Apply the TaskSingle pattern on Form.Design.

2. Address BP Warnings:
   b. The form must be referenced by at least one menu item.
   c. TabPage.Caption isn't empty.
   d. TabPage.DataSource isn't empty.
   e. StaticText.Text isn't empty.

**Related patterns**

- Task Double

**Commonly used subpatterns**

- Custom Filter Group
- Fields and Field Groups
- Toolbar and List
- Toolbar and Fields

**UX guidelines**

The verification checklist shows you the steps for manually verifying that the form complies with UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

**Standard form guidelines:**

- Standard form guidelines have been consolidated into the Microsoft Dynamics AX General Form Guidelines document.

**Task Single guidelines:**

- The Overview tab is the first tab and is active when the form is opened.
- The General tab must be the second tab and must have the label General.

**Examples**

Form: LedgerJournalTable
Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

- None

AX 2012 content
<table>
<thead>
<tr>
<th>Name</th>
<th>Journal batch number</th>
<th>Description</th>
<th>Posted</th>
<th>Log</th>
<th>In Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenJrn</td>
<td>000423_010</td>
<td>General Journal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GenJrn</td>
<td>000424_010</td>
<td>General Journal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GenJrn</td>
<td>000426_010</td>
<td>General Journal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GenJrn</td>
<td>000427_010</td>
<td>General Journal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trv</td>
<td>000428_010</td>
<td>Expense Journal</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This article provides information about the Wizard form pattern. A wizard is a special form of user assistance that takes the user through a task by using an ordered series of tab pages.

Usage

A wizard is a special form of user assistance that takes the user through a task by using an ordered series of tab pages. Wizards are especially useful for complex or infrequent tasks that the user might have difficulty learning or doing, or for tedious, frequently performed tasks.

Wireframe

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The secondary instruction for a wizard step was previously defined in the Help Text property of that step's Tab Page. This instruction will now be modeled on the Tab Page as a Static Text control.

Model

High-level structure

- Design (Style=Wizard; Caption=<wizard title>)
  - WizardContent (Tab)
    - WizardContentPage (TabPage) [repeats 1..N times, can be named anything; Caption set to page title]
    - MainInstruction (StaticText)
Core components

1. Apply the Wizard pattern on Form.Design.
2. Address BP Warnings:
   b. The form must be referenced by at least one menu item.
   c. TabPage.Caption isn't empty (for all wizard content pages).
   d. MainInstruction.Text isn't empty (for all wizard content pages).

Commonly used subpatterns

- Fields and Field Groups
- Toolbar and List
- Toolbar and Fields
- List Panel

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

Standard form guidelines:

- Standard form guidelines have been consolidated into the Microsoft Dynamics AX General Form Guidelines document.

Wizard guidelines:

- Each tab page should have a title.
- Each tab page should have a main instruction.
- Content should be subdivided into logical groups per page.
- A wizard should have <Next> and <Previous> buttons on the appropriate pages.
- The user should also be able to cancel the wizard, and cancellation should return to the state that existed before the wizard was started.
- Only one question should be asked per wizard page (tab page).
- When a set of choices is presented to the user, radio buttons should be used to make the alternatives clear, even if a check box or combo box is otherwise acceptable.
- Wizard forms must not have these elements:
  - FactBoxes
  - FastTabs

Examples

Form: WrkCtrBulkResReqEditWizard
Appendix

Frequently asked questions
This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

- None

AX 2012 content

AX 2012 links

- MSDN Wizards in Microsoft Dynamics AX [AX 2012]
- MSDN Guidelines for Wizard Development [AX 2012]

AX 2012 example
This topic discusses workspace form patterns. Workspaces are the primary way that users navigate to tasks and specific pages. A workspace should be created for every significant business activity that is supported.

**Usage**

Workspaces are a new concept, and are meant to be the primary way that users navigate to tasks and specific pages. A workspace should be created for every significant business “activity” that you want to support. An “activity” is less granular than a task and more granular than a legacy “area page.” A workspace is intended to provide a one-page overview of the activity, and to help users understand the current status, the upcoming workload, and the performance of the process or user. Users should be able to start the most typical tasks for the activity directly from the workspace. If possible, users should also be able to complete tasks directly in the workspace, based on the overview that they just received. Currently, there are two workspace patterns:

- **Tabbed workspace**: Instead of forcing a horizontally-scrolling panorama for content, this pattern uses standard tabs to allow the development of vertically-scrolling workspaces. This is particularly being used to embed Power BI reports into workspaces. Additional subpatterns to help define content inside these tabs will likely be provided in the future.

- **Operational workspace**: This is the standard pattern currently used for workspace development. Because of the set of components that are permitted in it, this pattern has superior performance over the deprecated “workspace” pattern. For this reason and to ensure visual and behavioral consistency with the other workspaces in the system, we recommend that you use this pattern.

- **(Deprecated) Workspace**: This pattern is only mentioned for the sake of completeness. Do not use this pattern. It will soon be removed from the product.

The rest of this topic will focus on the Operational workspace pattern and the Tabbed Workspace pattern, as the original Workspace pattern is deprecated and should not be used.

**Wireframe**

*Operational workspace*
Pattern changes for Finance and Operations
The Microsoft Dynamics AX 2012 Role Center has been replaced by multiple activity-focused workspaces.

Model

Operational workspace – High-level structure

- Design
  - Action pane (ActionPane) [Optional]
  - Workspace page filter group (Group) [Optional] – This must use the Workspace Page Filter Group subpattern.
  - Panorama (Tab)
- Section summary tiles (TabPage) – This must use the Section Tiles subpattern.
- Section tabbed list (TabPage) – This must use the Section Tabbed List subpattern.
- *Section charts (TabPage) [Optional]* – This must use the Section Stacked Chart subpattern.
- *Section PowerBI (TabPage) [Optional]* – This must use the Section PowerBI subpattern.
- Section related links (TabPage) – This must use the Section Related Links subpattern.

**Tabbed workspace – High-level structure**

- Design
  - *Action pane (ActionPane) [Optional]*
  - *Workspace page filter group (Group) [Optional]* – This must use the Workspace Page Filter Group subpattern.
  - StandardTab (Tab)
    - ContentTabPage (1..N)

**Core components**

- Apply the appropriate Workspace pattern on Form.Design.
- Address BP Warnings:
  - Form must be referenced by at least one menu item.
  - TabPage.Caption isn’t empty (for all panorama sections).

**Commonly used subpatterns**

- Workspace Page Filter Group
- Section Tiles
- Section Tabbed List
- Section Stacked Chart
- Section PowerBI
- Section Related Links

**Related patterns**

- Form Part Section List
- Section Chart

**UX guidelines**

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

- Standard form guidelines
  - Standard form guidelines have been consolidated into the General Form Guidelines document.
- Workspace form guidelines
  - Use a noun phase for the page title, and avoid general words. The page title should not duplicate the title of an area page.
  - The page title should begin with the noun that users would have in mind.
  - All sections must have a title.
  - A section typically spans the width of two to four standard tiles.
Any section that uses a FormPartControl to display content should have **HeightMode** set to **SizeToAvailable** on the FormPartControl.

- **Actions**
  - Include only frequently used commands.
  - Actions on the Action Pane should be related to the whole workspace (not a specific section of it).
  - **Exception:** A single "New" action can be put as a tile in the **Summary** section if it's very frequently used.
  - Group variations of the same command on drop-down menus.
  - **Examples:** New sales quote, New sales order, New return order

- **Filters**
  - Zero to five filter fields are allowed on a workspace.
  - Only a single field can be put under the page title
  - The remaining filters must be in a workspace configuration dialog.

**Example**

**Operational workspace**

**Form:** **FMClerkWorkspace**

![Image of a workspace](image)

**Frequently asked questions**

This section will have answers to frequently asked questions that are related to this guideline/pattern.

**Open issues**

- None

**AX 2012 content**

**AX 2012 links**

- [MSDN Role Center Page Reference](https://docs.microsoft.com/en-us/dynamicsax-2012/role-center-page-reference)
- [MSDN Role Center User Experience Guidelines](https://docs.microsoft.com/en-us/dynamicsax-2012/role-center-user-experience-guidelines)

**AX 2012 example**
### Common
- Projects
- Project resources
- Expenses
- Timecards
  - All timecards
  - My timecards
  - Unprinted timecards
  - Timecards for any review
  - Updated timecards history
- Transactions
- Project invoices
- Guidelines
- Grants

### Inquiries
- Forecasts
- Transactions
- Resource scheduling
- Committed costs
- Retention

### Periodic
- Invoices
- Forecasts
- Budgets
- Transactions
- Committed costs
- Project invoices
- Timecards
- Time and material
- Invoices
- Contracts

### Journals
- Journal
- Expense
- Item
- Fox
- Beginning balance

### Reports
- Date
- Transactions
- Resource scheduling
- Project statements
- Cash flow
  - Project cash flow
    - Category cash flow
    - Non-project cash flow
  - Timecards
  - Hour utilisation
  - Project invoices
  - Committed costs
  - Payroll utilisation
  - On-account
  - Estimated
  - Retained

### Setup
- Project management and accounting parameters
  - Project management and accounting workflow
  - Projects
  - Invoices
  - Transactions
  - Timecards
  - Time and material
  - Invoices
  - Contracts
  - Direct costs
  - Indirect costs
This article provides information about the Advanced Selection form pattern. This Dialog form pattern lets users filter and select items from a large, wide list. Like the List Panel pattern, this pattern should be used when the primary user task is to select a set of items.

**Usage**

The Advanced Selection form pattern should be used when the primary user task is to select a set of items. This task is usually accomplished through a multi-select list. However, in many scenarios, users must select items that aren’t contiguous and, at the same time, must see the set of items that they are selecting. This pattern resembles the List panel pattern, in that the user selects items in one list and adds them to another. However, this pattern allows for custom filters and a “wide” list on top, and uses most of the screen “real estate” of the page (typically, it’s a Large dialog). Use this pattern when a user must be able to filter and select in a large, wide list.

**Wireframe**

![Dialog wireframe](image)

**Related patterns**

- List Panel subpattern
- Dialog form pattern

**UX guidelines**

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.
- **Standard form guidelines:**
  - Standard form guidelines have been consolidated into the General form guidelines document.

- **Advanced selection guidelines:**
  - By default, the Quick filter should use the name or description column.
  - The list can display up to 15 columns. **Note:** This guidelines has been relaxed since Microsoft Dynamics AX 2012.
  - The main instruction should instruct users what they need to do.
  - When there is no data, the grid should not automatically add a new record.

**Example**

Form: `ProcCategoryAddVendor` (Click Procurement and sourcing > Procurement categories. On the Vendors FastTab, click Add.)

[NOTE] This form no longer utilizes this pattern; however, the image shows an example of what a typical Advanced selection form pattern looks like.
This topic provides information about the Dialog form pattern. A dialog box represents an action or activity that users can explicitly commit or cancel. It's used when a user initiates a specific task or process, and the system requires user input about how or whether to proceed.

Usage

A dialog box represents an action or activity that users can explicitly commit or cancel. It's used when a user initiates a specific task or process, and the system requires user input about how or whether to proceed. Dialogs are modal and require that users interact with the controls in the dialog before they can return to the parent page. Dialogs also can have multiple sizes. Selection of a dialog size is subjective, and will vary, depending on the form elements that you've modeled on the dialog. The sizes are as follows:

- **Small** – This size is a one-column-wide dialog. If your dialog contains a relatively small amount of content (all simple fields, and no wide tables or other wide elements), you can probably use this size.
- **Medium** – This size is a two-column-wide dialog. If your dialog contains more content than can comfortably fit within a small dialog, but a full-width dialog isn’t required, you should use this size.
- **Large** – This size is a three-column-wide dialog. If your dialog contains more content than can comfortably fit within a medium dialog, but a full-width dialog isn’t required, you should use this size.
- **Full** – A large dialog is nearly the full width of the browser viewport. Its size varies, depending on the viewport width, and it will always be the largest dialog size option. Use this size if your dialog has a lot of wide elements, or if it requires an unusually large amount of horizontal space.

For more detail about the various dialog sizes, see the table in the appendix of this topic, under “Selecting the correct dialog size.” We strongly recommend that you review that table. Five patterns are described in this document:

- **Dialog** – This is the basic dialog pattern. Use this dialog if you don’t have a reason to use one of the other Dialog patterns.
- **Dialog w/tabs** – This is a more specific version of the Dialog pattern. It incorporates a Tab control in the dialog. You can also optionally provide a header for the Tab, and also a footer.
- **Dialog w/FastTabs** – This closely resembles the Dialog w/tabs pattern but uses FastTabs instead of regular tabs to organize the information.
- **Dialog w/double tabs** – This closely resembles the Dialog w/tabs pattern but has a second Tab control immediately after the first one.
- **Dialog (read only)** – This pattern is for informational forms that aren’t editable. The user can still switch between tabs or a view selector, but direct manipulation of input fields isn’t allowed. This dialog variation also includes a Close button instead of OK and Cancel buttons.

Wireframe

The following sections show the wireframes for the four dialog types that are included in this article.
### Dialog

**Form caption**

- `SecondaryInstruction (0..1)`
- `ActionPane (0..1)`
- `DialogHeader (0..*)`

**DialogContent**

**DialogCommitContainer**

- `OKButton`
- `OtherButtons (0..*)`
- `CancelButton`

### Dialog w/tabs and Dialog w/FastTabs

**DialogTabs**

**Form caption**

- `SecondaryInstruction (0..1)`
- `ActionPane (0..1)`
- `DialogHeader (0..*)`

**TabContent**

**DialogFooter (0..1)**

**DialogCommitContainer**

- `OKButton`
- `OtherButtons (0..*)`
- `CancelButton`

### Dialog w/double tabs
**DialogDoubleTabs**

**Form caption**

*SecondaryInstruction (0..1)*

*ActionPane (0..1)*

*DialogHeader (0..* )*

TabContent

*ActionPane (0..1)*

TabContent

*DialogFooter (0..1)*

**DialogCommitContainer**

| OKButton | OtherButtons (0..* ) | CancelButton |

**Dialog (read only)**
### DialogReadOnly

<table>
<thead>
<tr>
<th>Form caption</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SecondaryInstruction (0..1)</strong></td>
</tr>
</tbody>
</table>

| ActionPane (0..1) |

| DialogHeader (0..*) |

| DialogContent |

| DialogCommitContainer |

| CloseButton |

### Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The form caption serves as the MainInstruction. Therefore, a modeled MainInstruction control is no longer required.
- Manual handling of error messages is no longer required.
- In some cases, buttons will scroll off the bottom of the slider if the form content exceeds the available height.
- Slider dialogs have their own message bar (because the main message bar is obscured when a slider is open).

### Model

**Dialog (basic) – High-level structure**

- Design
  - **SecondaryInstruction (StaticText) [Optional]**
    - **ActionPane (ActionPane) [Optional]**
  - **DialogHeader (Group, can repeat) [Optional]**
  - **DialogContent (Group, repeats 1..N)**
  - **DialogCommitContainer (ButtonGroup)**
    - **OKButton ($Button)**
    - **OtherButton ($Button, can repeat) [Optional]**
    - **CancelButton ($Button)**
Dialog w/Tabs and Dialog w/FastTabs – High-level structure

- Design
  - Secondary/Instruction (StaticText) [Optional]
  - ActionPane (ActionPane) [Optional]
  - DialogHeader (Group, can repeat) [Optional]
  - TabContent (Tab)
    - TabPage (TabPage, repeats 1..N)
  - DialogFooter (Group) [Optional]
  - DialogCommitContainer (ButtonGroup)
    - OKButton ($Button)
    - OtherButton ($Button, can repeat) [Optional]
    - CancelButton ($Button)

Dialog w/double tabs – High-level structure

- Design
  - Secondary/Instruction (StaticText) [Optional]
  - ActionPane (ActionPane) [Optional]
  - DialogHeader (Group, can repeat) [Optional]
  - TabContent (Tab)
    - TabPage (TabPage) [1..*]
  - TabContent (Tab)
    - TabPage (TabPage) [1..*]
  - DialogFooter (Group) [Optional]
  - DialogCommitContainer (ButtonGroup)
    - OKButton ($Button)
    - OtherButton ($Button, can repeat) [Optional]
    - CancelButton ($Button)

Dialog (read only) – High-level structure

- Design
  - Secondary/Instruction (StaticText) [Optional]
  - ActionPane (ActionPane) [Optional]
  - DialogHeader (Group, can repeat) [Optional]
  - DialogContent (Group, repeats 1..N)
  - DialogCommitContainer (ButtonGroup)
    - CloseButton ($Button)

Core components

- Apply the Dialog pattern on Form.Design.
- Address BP Warnings:
- **Design.Caption** isn't empty.
- The form must be referenced by at least one menu item.
- **StaticText.Text** isn't empty.

### Related patterns

- **Drop Dialog**

### Commonly used subpatterns

- **Fields and Field Groups**
- **Toolbar and List**
- **Toolbar and Fields**
- **Fill Text**

### UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps. **Standard form guidelines:**

- Standard form guidelines have been consolidated into the [General Form Guidelines](#) document.

### Dialog guidelines:

- Focus should be in the first editable field in the dialog box when the dialog box is first opened.
  - **Exception:** If the dialog is read-only, focus should be on the **Close** button.
- A dialog must have a **main instruction** at the top.
  - A final period should not be included if the instruction is a statement. If the instruction is a question, a question mark should be included.
- A secondary instruction to the user can optionally be included, and it should present additional information that will help the user understand or use the dialog box. The secondary instruction should consist of a complete sentence in sentence case and should have end punctuation.
- A dialog must have a **content** area.
- For editable dialogs:
  - The content area should contain only the controls that are required in order to complete the task.
  - Constrained input controls, such as selection lists, check boxes, radio buttons, and command links, should be used to avoid validation errors.
  - Reasonable default values for each input should be provided whenever possible.
  - Controls should not cause another dialog to appear during validation.
    - Warning messages for validation issues should be displayed in a message bar as soon as possible.
- For read-only dialogs:
  - The content area should only contain controls that are non-editable or only allow the user to switch the data is displayed, such as a view selector.
  - Programmatically changing the value of a field should not cause validation errors.
  - If your dialog has multiple tabs, the tab that has the most content must define the selection of the dialog width.
- A dialog must have a **commit** button area:
  - For an editable dialog only, there is a commit button that starts the action that is implied by the main instruction.
  - The labels should make sense on their own and should be a response to the main instruction.
  - For both editable and read-only dialogs, the right-most button is a **Cancel** button that cancels the
operation without side-effects.

- There is a button that is marked as the default button for the dialog.
- The button that is selected as the default button should be the safest, most secure response to the task that the user is performing, such as the main instruction of a Dialog or Drop Dialog.
- If safety and security aren’t factors, the button that is most likely to be clicked or that is most convenient for the user should be selected as the default button.
- **Exception:** Don’t select a destructive response as the default unless there is an easy, obvious way to undo the command.

A dialog should **not** have these elements:

- FactBoxes

**Examples**

**Dialog (basic)**

Form: ProjTableCreate (Click Project management and accounting > Common > Projects > All projects, and then click New.)

**Dialog w/tabs**

Form: CaseDetailCreate (Click Common > Common > Cases > All cases, and then click New.)
Dialog w/FastTabs

This example shows a modified version of the CaseDetailCreate form, because the product currently includes no examples of forms that use this pattern.

Dialog w/double tabs

Form: PurchTableReferences (Click Accounts payable > Common > Purchase orders > All purchase orders, and then click General > Related information > Related orders.)
Dialog (read only)

Form: SalesTablePostings (Click Accounts receivable > Common > Sales orders > All sales orders, and then click General > Related information > Postings.)
Open issues

- **How does this pattern handle the More info link in dialogs?**
  - *More info* usage is assumed to be a custom pattern unless we have enough cases to justify the addition of a new pattern.

- **Should the pattern be modified to force OK/Cancel buttons to use CommandButtons instead of any button type?**
  - We will be looking at making this change in the future.

### Selecting the correct dialog width

<table>
<thead>
<tr>
<th>TYPE OF CONTENT</th>
<th>SMALL DIALOG</th>
<th>MEDIUM DIALOG</th>
<th>LARGE DIALOG</th>
<th>FULL DIALOG</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns of content</td>
<td>The slider fits one column of content.</td>
<td>The slider fits two columns of contents.</td>
<td>The slider fits three columns of content.</td>
<td>The slider fits the viewport width minus peek.</td>
<td>The maximum number of columns depends on the width of the fields in the column. Therefore, the width is defined as $x \times 100%$ field size.</td>
</tr>
<tr>
<td>Horizontal scroll</td>
<td>No horizontal scrolling.</td>
<td>Avoid horizontal scrolling.</td>
<td>Avoid horizontal scrolling.</td>
<td>OK, provided that the control buttons and commit buttons are visible</td>
<td>Avoid putting so much content in the dialog that you cause vertical scrolling of the contents. If your dialog is vertically scrolling at a typical screen resolution, you should make the dialog larger.</td>
</tr>
<tr>
<td>Vertical scroll</td>
<td>No vertical scroll for typical scenarios (FastTabs can be expanded for special cases). Otherwise, use a Medium dialog.</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Switching tabs or expanding FastTabs should never cause jumps in the dialog size. The largest tab content must define the choice of the dialog size.</td>
</tr>
<tr>
<td>FastTabs</td>
<td>Strongly discouraged</td>
<td>OK but discouraged</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Tabs</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>TYPE OF CONTENT</td>
<td>SMALL DIALOG</td>
<td>MEDIUM DIALOG</td>
<td>LARGE DIALOG</td>
<td>FULL DIALOG</td>
<td>NOTES</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------</td>
<td>---------------</td>
<td>--------------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>List/hierarchy</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Field groups</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Nested field groups</td>
<td>No nested field groups that have a mixed layout direction (matrix or tabular layout)</td>
<td>Multiple-column layout</td>
<td>Multiple-column layout</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Custom controls</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Grid</td>
<td>Yes, but no horizontal scrolling</td>
<td>Yes, but no horizontal scrolling</td>
<td>Yes, but no horizontal scrolling</td>
<td>Yes</td>
<td>The maximum number of columns depends on the width of the fields in the column. Therefore, the width is defined by ( x \times 100% ) field size.</td>
</tr>
</tbody>
</table>

AX 2012 content

![Create project (1 - ceu) - New Record](image)

New project

- Project type: Time and material
- Project ID:
- Project name:
- Project group:
- Project contract ID:
- Customer:

[OK] [Cancel]
This topic provides information about the Drop Dialog form pattern. This pattern is used to initiate actions when the number of fields is seven or fewer.

Usage

The Drop Dialog pattern is used to initiate actions when the number of fields is seven or fewer. Drop dialogs are quick and easy for users to use, and are more lightweight than a full dialog that is presented as a slider. Drop dialogs should feel as lightweight to use as a menu. Two patterns are described in this document:

- **Drop dialog** – This is the basic Drop dialog pattern. If your Drop dialog is editable, this is the correct pattern to use.
- **Drop dialog (read only)** – This Drop dialog pattern is for informational forms that aren't editable. This variation doesn't have an OK button.

Wireframe

**Drop dialog (basic)**

Drop Dialog

**Form caption**

*Secondary instruction (optional)*

**DialogContent**

**DialogCommitContainer**

**OKButton**

**Drop dialog (read only)**
**Pattern changes**

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- Manual handling of error messages is no longer required.

**Model**

**Drop dialog (basic) – High-level structure**

- Design
  - `SecondaryInstruction (StaticText) [optional]`
  - DialogContent (Group)
  - DialogCommitContainer (ButtonGroup)
    - OKButton ($Button)

**Drop dialog (read only) – High-level structure**

- Design
  - `SecondaryInstruction (StaticText) [optional]`
  - DialogContent (Group)

**Core components**

- Apply the Drop Dialog pattern on `Form.Design`.
- Address BP Warnings:
  - `Design.Caption` isn't empty.
  - The form must be referenced by at least one menu item.
  - `StaticText.Text` isn’t empty.

**Related patterns**

- Dialog
Commonly used subpatterns

- Fields and Field Groups
- Toolbar and List

UX guidelines

The verification checklist shows you the steps for manually verifying that the form complies with UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps. **Standard form guidelines:**

- Standard form guidelines have been consolidated into the [General Form Guidelines](#) document.

**Drop dialog guidelines:**

- A Drop dialog should be used if the following conditions exist:
  - There are seven or fewer fields.
  - The user can enter the information quickly.
  - Minimal field validation is required.
  - There are no buttons that open additional child forms.
    - **Exceptions:** Lookups, Enhanced preview, and View details navigation
  - There is no editable grid (select-only grids are allowed).
- Focus should be in the first editable field on the Drop dialog when it is first opened.
- A Drop dialog should have a **main instruction** (form caption) at the top.
  - The main instruction should be used to explain concisely what the user should do in the Drop dialog. The instruction should be a specific statement, an imperative direction, or a question. Good instructions communicate the user’s objective with the Drop dialog rather than focusing purely on the mechanics of manipulating it.
  - A final period should not be included if the main instruction is a statement. If the instruction is a question, a question mark should be included.
  - Besides the main instruction, a secondary instruction to the user should be displayed, and it should present additional information that will help the user understand or use the Drop dialog. The secondary instruction should consist of a complete sentence in sentence case and should have end punctuation.
    - **Exception:** If the additional instruction merely repeats the main instruction with slightly different wording, don’t include it.
- A Drop dialog should have a **content** area.
  - Constrained input controls should be used to avoid validation errors. Examples include selection lists, check boxes, radio buttons, and command links.
  - Reasonable defaults for each input should be provided whenever possible.
- A Drop dialog should have a **commit** button area that:
  - Does **not** have a **Cancel** button.
  - Has a button that is marked as the default button of the Drop dialog (if a button exists).
  - The label of the default button should be a verb that implements the action that is described in the main instruction. For example, if the main instruction is “Create new product,” the button label should be **Create**. If there is no appropriate verb for the button, use **OK**.
  - The commit button area should have specific commit button labels that make sense on their own and are a response to the main instruction.
- A Drop dialog should **not** have the following:
  - A toolbar or ActionPane anywhere in the Drop dialog.
• Buttons that navigate to another page or open other dialogs. (Enhanced previews are allowed.)
• Field groups. There are exceptions, such as a radio button or check box group.
• A tab control.
• FactBoxes.
• FastTabs.

Examples

Drop dialog (basic)
Form: CustCollectionsNewActivityAction (Click Accounts receivable > Common > Collections > Collections, select a row to move to details, and then click Action.)

Enter an activity for a completed action

Type

Purpose

Notes

Date and time closed

3/17/2015 08:28 PM

Create action

Drop dialog (read only)
This pattern isn’t currently used in the product.

Appendix

Frequently asked questions
This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues
• Should a vertical Fields and Field Groups subpattern be added for Drop dialogs?
  ○ No, you should use the normal Fields and Field Groups pattern.
• Should buttons be left-aligned or right-aligned?
  ○ Right-aligned. The pattern is currently enforcing this.

AX 2012 content
Enter an activity for a completed action

<table>
<thead>
<tr>
<th>Type</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td></td>
</tr>
<tr>
<td>Notes</td>
<td></td>
</tr>
</tbody>
</table>

Date and time closed: 12/1/2006 03:26 pm

Create action
This topic provides information about the Lookup form pattern. Custom lookup forms should be used when a standard framework-provided lookup would not provide the correct data, or when advanced visualization of the data is required.

Usage

Custom lookup forms should be used when a standard framework-provided lookup (which is typically generated by using the AutoLookup field group that is defined on the table definition), would not provide the correct data, or when advanced visualization of the data is required. Three patterns are described in this document:

- **Lookup basic** – This is the basic Lookup pattern that has just one list or tree, and also optional custom filters and actions.
- **Lookup w/tabs** – This Lookup pattern is used when more than one view of the lookup can be made available to the user. Tab captions aren’t shown. Instead, the tab is selected through a combo box.
- **Lookup w/preview** – This more advanced Lookup pattern enables a preview of the current record in the lookup grid.

Wireframe

**Lookup basic**

- **Lookup**
  - **Custom filters (optional)**
  - **Grid/Tree/ListView**
  - **Lookup actions (optional)**

**Lookup with tabs**
Pattern changes

Here are the changes to this pattern since Microsoft Dynamics AX 2012:

- Tabs should be hidden and controlled by a combo box.
- Optionally add a splitter/preview.

Model

Lookup basic – High-level structure
- Design
  - CustomFilter (Group) [Optional]
  - Grid | Tree | ListView
  - LookupActions (Group) [Optional]

**Lookup w/tabs – High-level structure**
- Design
  - CustomFilter (Group) [Optional]
  - LookupTab (Tab)
    - LookupTabPage (TabPage, repeats 1..N)
      - Grid | Tree | ListView
    - LookupActions (Group) [Optional]

**Lookup w/preview – High-level structure**
- Design
  - CustomFilter (Group) [Optional]
  - LookupContent (Group)
    - Grid | Tree | ListView
    - VerticalSplitter (Group)
      - Preview (Group)
      - LookupActions (ActionPane)

**Core components**
- Apply the Lookup pattern on Form.Design.
- Address BP Warnings:
  - EDT.FormHelp must reference a form where Style=Lookup.

**Commonly used subpatterns**
- Custom Filter Group

**UX guidelines**
The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in a browser, and walk through these steps. **Standard form guidelines**

- Standard form guidelines have been consolidated into the General Form Guidelines document.

**Lookup guidelines**
- Grid guidelines have been consolidated into the General Form Guidelines document, in the Grid guidelines section.
- If you must show different “views” (tabs) within the lookup, use a combo box to let the user to switch between tabs.
- You can optionally use a tree view in the lookup. Also consider providing a standard grid because of the complexity that is involved in showing additional fields of data in a tree.
- Don’t have more than five columns in the grid. The lookup resizes to show all columns, so five columns is very wide.
The optional preview area:
- The area should help the user choose between two or more records that are similar. For example, if you have two employees who are named John Smith, the preview should provide enough information to help the user differentiate these two people.
- Don't show editable fields in the preview.

Custom filter guidelines have been consolidated into the Custom Filter Group subpattern document.

Examples

Lookup basic
Form: SysLanguageLookup (Click Settings > User settings on the navigation bar)

Lookup with tabs
Form: CaseCategoryLookup (Click Common > Common > Cases > All cases, and then select a case to go to the details.)

Lookup with preview
Form: HcmWorkerLookup (Click Human resources > Common > Organization > Positions > Positions, and then click a record to go to the details. Expand the Worker assignment FastTab, click New, and then click the drop-down arrow in the Worker field.)

Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

- **How do I switch between tabs in the Lookup w/tabs pattern?**
  - The Lookup w/tabs pattern intentionally sets ShowTabs to No on the Tab control. These forms are meant to model an unbound combo box in the custom filter group. This combo box is used instead of tab headers to switch tabs.
  - To do this, follow these steps:
    1. Call the SysLookup::tab2ComboBox method post super in form run() to populate the combo box with captions from visible tabs in the lookup.
    
       ```
       // Generate view combobox based on tabs
       tab2ComboBoxItemMap = SysLookup::tab2ComboBox(Tab, switchView);
       ```
    2. Override modified() on the combo box to update the visible tab, based on the selected value in the combo box.
    
       ```
       Tab.tabChanged(Tab.tabValue(), tab2ComboBoxItemMap.lookup(this.selection()));
       ```

Open issues

- **Can we incorporate the most recently used values into lookups?**
  - App modeling can make this work right now (for example, the Currency lookup). We are considering general framework support for this feature in the future.

AX 2012 content

SysLanguageLookup (Lookup basic)
CaseCategoryLookup (Lookup with tabs)

HcmWorkerLookup (Lookup with preview)
This topic provides information about the FactBox form patterns. FactBoxes are used to provide related information for a record.

Usage

In general, FactBoxes are used to provide "related information" for a record. They help guarantee that the user doesn't have to open additional forms to get important information, such as totals, balances, overdue orders, and email addresses. The Factbox Grid pattern should be used when there is a child collection (potential for multiple rows) of related information. Two patterns are described in this document:

- **Form Part FactBox Grid** – This FactBox pattern is used when there is a child collection (potential for multiple rows) of related information.

- **Form Part FactBox Card** – This FactBox pattern is used when there is just a set of related fields that must be shown.

Wireframe

**Form Part FactBox Grid**

FactBox

Grid

Button Group (optional)

Button

**Form Part FactBox Card**
Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- A group has been added around the optional button to make it easier to position the button.

Model

**Form Part FactBox Grid – High-level structure**

- Design
  - Grid
    - GridDefaultAction (Button) [Optional]
    - ButtonGroup (ButtonGroup) [Optional]
  - Button

**Form Part FactBox Card – High-level structure**

- Design
  - FieldGroups (Group) [0..N]
    - Fields ($Fields, 1..N)
    - Fields ($Field) [0..N]
    - ButtonGroup (ButtonGroup) [Optional]
  - Button

Core components

- Apply the FactBox pattern on Form.Design.
- Address BP Warnings:
  - Design.Caption isn't empty.
  - Grid.DataSource isn't empty.

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps. Standard form guidelines:
Standard form guidelines have been consolidated into the General Form Guidelines document.

FactBox general guidelines:

- If a backing form exists, the FactBox should have a (More…) link defined that goes to the appropriate backing form. The names of the FactBox and backing form should be similar.
- The title should not be a verb or a verb phrase.
- The title should not contain a label to a specific record.
- FactBoxes should not display fields that let a user enter data by typing with the keyboard.
- The title should accurately describe the content and should not be truncated when the FactBox area is at its default size.

FactBox grid guidelines:

- One to four columns should be displayed.

FactBox card guidelines:

- Each field should have a label.
- The ID and name of the header or the line that content is displayed for in the FactBox should not be displayed.
- Two to ten fields should be displayed.
- Currency indicator fields should be displayed as the last field in the FactBox.

Examples

Form Part FactBox Grid
Form: CustTable > ContactsInfoPart

Form Part FactBox Card
Form: CustTable > CustStatisticsStatistics
Appendix

Frequently asked questions
This section will have answers to frequently asked questions that are related to this guideline/pattern.

- **How do I make the More button work?**
  - The More button at the bottom of the FactBox takes the user to a backing form that contains the full list of related records. This button should be implemented by using a regular Button control that overrides the clicked method as shown in the following example. Be sure to fill in the TableRef and ListPageRef properties on the table that provides data for the grid.

```csharp
[Control("Button")]
class More
{
    public void clicked()
    {
        super();
        FormPartUtil::openShowMoreForm(element, <TableName>);
    }
}
```

Open issues
- **Should field labels be on the left side in FactBoxes to support a more compact visual?**
  - We plan to allow LabelPosition=Left inside FactBoxes.

AX 2012 content

AX 2012 links
- AX 2012 MSDN List Page Guidelines (including FactBoxes)

AX 2012 example

CustTable > ContactsInfoPart

![Example of FactBox with More button](image)
Usage

This subpattern is used to show a small collection of input controls (no more than five) that apply a custom filter to a grid or form section. Fields in the Custom Filter Group should be limited to the following field types, which have constrained inputs and can be applied to the query:

- StringEdits with Lookups
- Date fields
- ReferenceGroup
- Comboboxes
- Checkboxes
- Quick Filter

Two patterns are described in this document. The only difference between these patterns is whether the Quick Filter control is mandatory or optional:

- **Custom Filters** – In this subpattern, the QuickFilter control is optional.
- **Custom and Quick Filters** – In this subpattern, the QuickFilter control is mandatory.

Wireframes

Custom Filters

```
Group

Quick Filter (opt) Group (0..n) Field (0..n)

Field (1..n)
```

Custom and Quick Filters

```
Group

Quick Filter Group (0..n) Field (0..n)

Field (1..n)
```

Model

**Custom Filters – High-level structure**

- CustomFilter (Group)
  - QuickFilter (QuickFilter) [Optional]
  - FieldGroups (Group) [0..N]
Custom and Quick Filters – High-level structure

- CustomFilter (Group)
  - QuickFilter (QuickFilter)
  - FieldGroups (Group) [0..N]
    - Fields ($Field) [1..N]
    - Fields ($Fields) [0..N]

Core components

- Apply a custom filter-container pattern to a Group control.
- Address BP Warnings:
  - Input controls within the CustomFilterGroup should not have a DataSource or DataField assigned.

Related container patterns

None

Related modeling

Use QueryFilter, not QueryBuildRange, for all custom filters. QueryBuildRange doesn’t work correctly with outer-joined fields.

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in a browser, and walk through these steps.

- Standard form guidelines:
  - General form guidelines

- Custom Filter Group guidelines:
  - All controls (except QuickFilter) are constrained input controls. Open-ended controls, such as strings, integers, and reals, should not be used.
  - Field labels are turned off to save space. For example, no label is needed for a combobox that has the values Open, Closed, Posted, and All.
    - Show labels when the filter values do not provide sufficient context for the user to understand what the filter does. For example, a date field provides no context, and the user requires a label to understand what type of date is specified (for example, Created date).
    - Either all labels are turned off, or all labels are turned on. Don’t mix unlabeled filters and labeled filters.
    - Exception: When you use a check box–style Boolean, the label can be left on, even though other fields don’t show a label.
    - There should not be more than five controls in the custom filter group.

Examples

Custom Filters

Form: LedgerJournalTable (TopFields)
Custom and Quick Filters

Form: CustTable (CustomFilterGroup)

Resources

Typically used by form patterns
- Simple List
- Details Master
- Details Transaction
- List Page

Appendix

Frequently asked questions
This section will have answers to frequently asked questions that are related to this guideline/pattern.

- **What do I do with the legal entity?**
  - “Legal entity” is a typical custom filter that belongs in the Custom Filter Group.

- **Should the custom filter be above the toolbar of a list?**
  - We think that the custom filter belongs as close to the grid as possible, because it more directly affects the list, just as the commands in the toolbar do. Additionally, this position makes the logical order of the elements consistent across different page patterns.

Open issues
- **Do we allow Show More/Less in the Custom Filter Group? An example is BudgetAnalysisInquiry_PSN.**
  - No, that will currently be a custom container. If we have enough examples, we might add a new container subpattern.

- **Does the pattern limit the possible input types to those that allow constrained input values?**
- The pattern currently allows any input, but it's against guidelines to have inputs that have unconstrained values.

- **Do we allow groups to be used as custom filter groups?**
  - We do allow them now, to make migration easier and to identify Custom Filter group locations. However, we encourage you to use only a small set of fields in these situations (and we might eventually enforce this).

**AX 2012 content**

**AX 2012 links**

- [MSDN AX 2012 How to Add Controls to the Filter Pane](#)
- [MSDN AX 2012 List Page Overview – section Filter Pane](#)

**AX 2012 example**

![Example of a list page with filter pane controls](image-url)
This article provides information about the Dimension Entry Control subpattern. This subpattern is used when you have a group or tab page that uses the Dimension Entry control (DEC).

Usage

The Dimension Entry Control pattern is used when you have a group or tab page that uses the Dimension Entry control (DEC).

Wireframe

```
Container

Top Field Group (optional)

Dimension Entry Control (0..n)

Group (0..n)

Dimension Entry Control

Bottom Field Group (optional)
```

Model

**High-level structure**

TabPage | Group TopFieldGroup (Group) [Optional] – **Note:** A field subpattern is used.

DECGroup (Group) [0..N] Dimension Entry Control Dimension Entry Control [0..N] BottomFieldGroup (Group) [Optional] – **Note:** A field subpattern is used.

**Core components**

- Apply the Dimension Entry Control subpattern to the TabPage control.

**UX guidelines**

None.

**Examples**
Appendix

Frequently asked questions
This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues
None.
This article describes the Dimension Expression Builder subpattern, which is applied to container controls that use the Dimension Expression Builder control.

Usage

The Dimension Expression Builder pattern is used when you have a group or tab page that uses the Dimension Expression Builder control.

Wireframe

![Wireframe Diagram]

Model

High-level structure

TabPage | Group

TopFieldGroup (Group) [Optional] – Note: A field subpattern is used.

DEBGroup (Group) [0..N]

Dimension Expression Builder

Dimension Expression Builder [0..N]

Core components

- Apply the Dimension Expression Builder subpattern to the TabPage or Group control.

UX guidelines

None

Examples

Form: BudgetControlConfiguration (RulesDetailsCriteriaFastTabPage) (Budgeting > Setup > Budget)
Appendix

Frequently asked questions
This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues
None
This topic provides information about the Field and Field Groups form subpattern. This is the most common data entry subpattern. It uses a dynamic number of columns to present multiple fields or groups of fields.

Usage

Field and Field Groups is the most common data entry subpattern and uses a dynamic number of columns to present multiple fields or groups of fields. This subpattern is not used with controls that have dynamic height or width (for example Grid, Tree, RadioButton, ListBox, or ListView), or controls that have larger height or width (for example, Chart). The group controls within this pattern can be used either to group fields under a label or to bind to a table field group.

Typical contents

- Groups or Fields as immediate children of the FastTab
- Groups containing Fields
- Can contain other subpatterns:
  - Horizontal fields and button group

Wireframe

![Wireframe diagram]

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- Removed explicit columns and the use of groups to force fields into two or three (or more) columns.
- Changed from fixed columns to dynamic columns.

Model

**High-level structure**

- [Container] (Columns=Fill)
  - FieldGroups (Group) [0..N]
Fields ($Field) [1..N]
- ActionableFields (Group) [0..N] mimics the Horizontal Fields and Button Group subpattern
- Fields ($Field) [0..N]
- ActionableFields (Group) [0..N]

**Core components**
- Apply the FieldsAndFieldGroups subpattern to the container control.
- Address BP Warnings:
  - No additional BP checks are required beyond the AX6.3 BP that were checks carried forward.

**Related patterns**
- Horizontal Fields and Buttons Group

**UX guidelines**

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in a browser, and walk through these steps.

- **Standard form guidelines:**
  - Standard form guidelines have been consolidated into the General Form Guidelines document.
- **Fields and Field Groups guidelines:**
  - The fields in groups should flow across the entire page.
  - When possible, remove unnecessary field group labels.
  - Verify that you have an understandable grouping for your fields.
  - Either all fields should be in Groups that have labels, or no Group labels should be shown.

**Examples**

*Form: InventLocation (LocationNames)*
Resources

Typically used by patterns
- Simple List and Details
- Table of Contents
- Details Master
- Details Transaction

Appendix

Frequently asked questions
This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues
- Tooling must allow explicit use of the HorizontalFieldsButtonsGroup subpattern instead of mimicking content in the pattern definition.

AX 2012 content

InventLocation
This topic provides information about the Filters and Toolbar subpatterns. These workspace-specific subpatterns have been developed to show filters and/or actions inside panorama sections that host lists and charts.

Usage

The Filters and Toolbar subpatterns are workspace-specific subpatterns that have been developed to show filters and/or actions inside panorama sections that host lists and charts. Fields in the filtering parts of these subpatterns should be limited to the following field types. All these field types have constrained inputs and can be applied to the query.

- StringEdits with Lookups
- Date fields
- ReferenceGroup
- Comboboxes
- Checkboxes
- Quick Filter

Two subpatterns are described in this article:

- **Filters and Toolbar - Inline** – In this subpattern, any defined actions appear on the same line as the filter fields.
- **Filters and Toolbar - Stacked** – In this subpattern, any defined actions appear on a separate line below the filter fields.

Wireframe

**Filters and Toolbar - Inline**

<table>
<thead>
<tr>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>FilterGroup (optional)</td>
</tr>
<tr>
<td>QuickFilter (optional)</td>
</tr>
<tr>
<td>Toolbar (optional)</td>
</tr>
</tbody>
</table>

**Filters and Toolbar - Stacked**
Model

Filters and Toolbar - Inline: High-level structure
- Group (ArrangeMethod=HorizontalLeft)
  - FilterGroup (Group) [Optional]
    - QuickFilter (QuickFilter) [Optional]
    - FilterFields ($Field) [0..N]
    - Toolbar (ActionPane) [Optional]

Filters and Toolbar - Stacked: High-level structure
- Group (ArrangeMethod=Vertical)
  - FilterGroup (Group) [Optional]
    - QuickFilter (QuickFilter) [Optional]
    - FilterField1 ($Field) [Optional]
    - FilterField2 ($Field) [Optional]
    - Toolbar (ActionPane) [Optional]

Core components
Apply the correct Filters and Toolbar subpattern to the container control.

Related container patterns
- Form Part Section List
- Section Chart

UX guidelines
The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

- Filters and Toolbar guidelines
  - The Stacked variant should be used over narrow lists and charts.
  - The Inline variant should be used over wider lists and charts.

- Filters
  - No more than two filter fields should be used in a Filter group. If you require more than two filter fields, an action on the Toolbar can be used to open a Drop Dialog that has more filter fields.
  - The filter fields should not have labels. The context should be obvious from the field value.
The combined width of filter fields should not cause the section to become larger than the grid or chart in the section, and should not cause an extra scrollbar on the filters.

- **Actions**
  - Include only frequently used commands that help users complete tasks in the workspace.
  - No more than three actions should appear on the Toolbar. One action on the Toolbar can be used as a drop-down list of up to three additional actions.

**Examples**

**Filters and Toolbar - Inline**

Form: `HcmWorkforceManagement > HcmOpenPositionsPart (All workspaces > Workforce management)`

<table>
<thead>
<tr>
<th>Position</th>
<th>Title</th>
<th>Department</th>
<th>Reports to</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>000022</td>
<td>Sales Associate - USA ...</td>
<td>Retail Operations</td>
<td>Ken Thomee</td>
<td></td>
</tr>
<tr>
<td>000114</td>
<td>Director of Human Res...</td>
<td>Human Resources</td>
<td>Charlie Carson</td>
<td></td>
</tr>
<tr>
<td>000134</td>
<td>Sales Associate - USA ...</td>
<td>Retail Operations</td>
<td>Mike Danseglio</td>
<td></td>
</tr>
<tr>
<td>000151</td>
<td>Sales Associate - USA ...</td>
<td>Retail Operations</td>
<td>Steve Fanch</td>
<td></td>
</tr>
<tr>
<td>000204</td>
<td>Sales Associate - USA ...</td>
<td>Retail Operations</td>
<td>Pedro Ferrisna</td>
<td></td>
</tr>
<tr>
<td>000227</td>
<td>Sales Associate - Europe</td>
<td>Retail Operations</td>
<td>Burke Fewel</td>
<td></td>
</tr>
<tr>
<td>000264</td>
<td>Dispatcher</td>
<td>Operations</td>
<td>Daniel Brunner</td>
<td></td>
</tr>
<tr>
<td>000287</td>
<td>Accounts Payable Coor...</td>
<td>Finance</td>
<td>Phyllis Harris</td>
<td></td>
</tr>
</tbody>
</table>

**Filters and Toolbar - Stacked**

Form: `HcmWorkforceManagement > HcmWorkerOnLeaveListPart (All workspaces > Workforce management)`

**Appendix**

**Frequently asked questions**

This section will have answers to frequently asked questions that are related to this guideline/pattern.

**Open issues**

Why does the Inline variant allow for an arbitrary number of filter fields, but the Stacked variant allows a maximum of three (a QuickFilter and two custom filters)?
Two factors contribute to this discrepancy:

- A UX guideline specifies a maximum of two filters in these sections (and one of those filters could be a QuickFilter). Therefore, the Stacked variant more closely complies with the guideline.
- The number of fields in the Stacked variant is limited for aesthetic reasons. The filter fields in this variant are intended to take up the full width of the list/chart that appears below them, and their width is therefore `SizeToAvailable`. When this variant is used above narrow lists, as it’s intended to be used, that setting can cause very narrow filter fields when more than two filter fields are used. The Inline variant is intended to be used above wider charts/lists. Therefore, the original pattern definition allowed for an arbitrary number of fields. Nevertheless, we do plan to address this discrepancy in the number of allowed filter fields between the two variations in the future.
This article provides information about the Fill Text subpattern. This subpattern is used when a single String or StaticText control must stretch to the full width of the container, so that users have more space to enter information.

Usage
Fill Text is used when you need a single String or StaticText control to stretch to the full width of the container. This subpattern is typically used for multi-line string controls that require more space for users to enter information.

Wireframe

<Any Container>
Field

Model

High-level structure
[Container]
String | StaticText

Core components
• Apply the Fill Text subpattern to the container control.

Related container patterns
• Fields and Field Groups

UX guidelines
None

Examples
Form: FmRental (Notes)
Resources

Typically used by patterns
- Details Master
- Details Transaction
- Simple Details
- Simple List and Details
- Table Of Contents
- Wizard

Appendix

Frequently asked questions
This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues
- The pattern currently sets the **HeightMode** property of the control to **SizeToAvailable**. This can produce very tall string controls if the pattern is used in a **SizeToAvailable** container. We're investigating whether this control should use **SizeToContent** height, or whether it should not set the property at all and should instead let the developer decide the appropriate control height.
This article provides information about the Horizontal Fields and Buttons Group form subpattern. This subpattern is used when actions must be defined for an individual field on a form.

Usage

This subpattern is used when actions must be defined for an individual field on a form. The buttons are laid out just to the right of the field to visually associate the actions with the field. The buttons should display only an icon (no text). Actions that are associated with a section or an entire form should be placed in a Toolbar or ActionPane above that section or form.

Typical contents

- 1–2 fields
- 1–3 buttons

Wireframe

```
Group
Field  Field (optional)  Button  …  Button (optional)
```

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- The layout of fields and buttons will use a single column, where `ArrangeMethod=HorizontalLeft`.

Model

High-level structure

- Group (ArrangeMethod=HorizontalLeft)
  - Field
  - Field (optional)
  - Buttons (1–3 buttons)

Core components

- Apply the HorizontalFieldsButtonsGroup subpattern to the container control.
- Address BP Warnings:
  - There should be no more than three buttons.
  - No additional BP checks are required beyond the AX6.3 BP checks that were carried forward.

Related patterns

- Toolbar and Fields
- Horizontal Fields

UX guidelines
The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in a browser, and walk through these steps.

- **Standard form guidelines:**
  - Standard form guidelines have been consolidated into the General Form Guidelines document.

- **Horizontal Fields and Buttons Group guidelines:**
  - The width of the fields + buttons should not exceed the standard size of a column.
  - Buttons should have a symbol image assigned.
  - Buttons should have tooltips.
  - There should be a maximum of three buttons. The last button can be a menu button.

### Examples

**Form:** SalesTable (GroupHeaderAddressHeaderOverview)

![Delivery Address Form](image)

### Resources

**Typically used by patterns**

- Simple List and Details
- Table of Contents
- Details Master
- Details Transaction

### Appendix

**Frequently asked questions**

This section will have answers to frequently asked questions that are related to this guideline/pattern.

**Open issues**

- None

**Dynamics AX 2012 content**

SalesTable
<table>
<thead>
<tr>
<th>Delivery address</th>
<th>Delivery date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: Peer Conference Center</td>
<td>Requested ship date: 7/27/2012</td>
</tr>
<tr>
<td>Delivery address: Peer Conference Center (After hours)</td>
<td>Requested receipt date: 7/27/2012</td>
</tr>
<tr>
<td>Address: 123 Apple Street Beaverton, OR 97070 US</td>
<td>Confirmed ship date:</td>
</tr>
<tr>
<td></td>
<td>Confirmed receipt date:</td>
</tr>
</tbody>
</table>
This article provides information about the Image Preview form subpattern. This subpattern can be used for most images that appear within a form container, especially within a FastTab or Group.

**Usage**

Image Preview can be used for most images that appear within a form container, especially within a FastTab or Group. This subpattern can be used in conjunction with the FieldsAndFieldGroup and FillText subpatterns to combine images and any associated fields. This subpattern isn't used for tiles or buttons, or for field status images.

**Typical contents**

- Toolbar (ActionPane where Style=Strip)
- Image
- Can contain subpatterns:
  - Fields and Field Groups
  - Fill text

**Wireframe**

![Image Preview Wireframe](image)

**Pattern changes**

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- Fields are to the right of the image, if there are any fields.
- An ActionPane above the image can be used for associated actions (for example, Upload and Select).

**Model**

**Image only – High-level structure**

- [Container] (Columns = Fixed – 1)
  - Toolbar (ActionPane) [Optional]
  - Image
Image and fields – High-level structure

- [Container] (Columns = Fixed – 1)
  - Toolbar (ActionPane) [Optional]
  - Image
  - Group
    - Image
    - Group - Note: uses a fields subpattern

Core components

- Apply the Image Preview subpattern to the container control.
- Address BP Warnings:
  - No additional BP checks are required beyond the AX6.3 BP checks that were carried forward.

Related container patterns

- Fields and Field Groups
- Fill Text

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with the UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in a browser, and walk through these steps.

- Image Preview guidelines:
  - Any fields should be placed to the right of the image.

Examples

Form: RetailVisualProfile (Login)

Resources

Typically used by patterns

- Details Master
- Details Transaction
- Simple Details
- Simple List and Details
- Table of Contents

Appendix

Frequently asked questions
This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues
None.

AX 2012 content
This article provides information about the List Panel form subpattern. Application teams use this subpattern to manage two lists that move data between each other.

Usage

List Panel is the subpattern that application teams use to manage two lists that move data between each other. This pattern is meant to represent a modeled version of the SysListPanel class (programmatic) approach of managing two lists that move data between each other. The List Panel subpattern can be applied on the following controls:

- TabPage control
- Group control

Wireframe

![List Panel Wireframe](image)

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- List (Grid/ListView) and Tree controls are supported.
- The right panel is the selected section.
- The left panel is the available section.
- Six buttons are available as actions:
  - Add
  - Remove
  - Add All (Optional)
  - Remove All (Optional)
  - Move Up (Optional)
  - Move Down (Optional)

Model

**High-level structure**

- [Container]
Core components
- Apply the ListPanel subpattern to the container (TabPage or Group) control.
- Address BP Warnings:
  - No additional BP checks are required beyond the AX6.3 BP checks that were carried forward.

UX guidelines
The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in a browser, and walk through these steps.

- **Standard form guidelines:**
  - Standard form guidelines have been consolidated into the General Form Guidelines document.

Examples
Form: **SalesSummaryParameters (GroupQuotation)**

<table>
<thead>
<tr>
<th>AVAILABLE</th>
<th>SELECTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct delivery</td>
<td>Invoice account</td>
</tr>
<tr>
<td>Original customer</td>
<td>Currency</td>
</tr>
<tr>
<td>Company</td>
<td></td>
</tr>
<tr>
<td>Due date</td>
<td></td>
</tr>
<tr>
<td>Payment specification</td>
<td></td>
</tr>
<tr>
<td>Method of payment</td>
<td></td>
</tr>
<tr>
<td>Warehouse</td>
<td></td>
</tr>
<tr>
<td>List code</td>
<td></td>
</tr>
</tbody>
</table>

Resources
**Typically used by patterns**
Appendix

Frequently asked questions
This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues
None

AX 2012 content

<table>
<thead>
<tr>
<th>Selected financial dimensions</th>
<th>Available financial dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>MainAccount</td>
<td>CustomDepartmentA</td>
</tr>
<tr>
<td></td>
<td>CustomCostCenterA</td>
</tr>
<tr>
<td></td>
<td>CustomPurposeA</td>
</tr>
<tr>
<td></td>
<td>CustomDepartmentB</td>
</tr>
<tr>
<td></td>
<td>CustomCostCenterB</td>
</tr>
<tr>
<td></td>
<td>CustomPurposeB</td>
</tr>
<tr>
<td></td>
<td>CustomDepartmentC</td>
</tr>
<tr>
<td></td>
<td>CustomCostCenterC</td>
</tr>
<tr>
<td></td>
<td>CustomPurposeC</td>
</tr>
<tr>
<td></td>
<td>CustomDepartment</td>
</tr>
</tbody>
</table>
This topic provides information about the Nested Simple List and Details (NSL+D) subpattern. This subpattern is used to display information about a secondary or child entity when that child entity is presented within another form type.

Usage

This article describes a variant of the Simple List and Details (SL+D) pattern that is named the Nested Simple List and Details (NSL+D) subpattern. Whereas the SL&D form pattern is used to display information about the primary entity on the form, the NSL+D subpattern is used to display information about a secondary or child entity when that child entity is presented within another form type. The amount of information that is related to the child entity should be too much for a grid (10 or more fields) but not enough for the child entity to deserve its own form. The NSL+D subpattern has a few differences from the SL+D form pattern:

- You may not nest an NSL+D subpattern within another NSL+D subpattern.
- The NSL+D subpattern uses a Toolbar for contextual actions.
- The details portion of the NSL+D subpattern is simpler than the SL+D pattern. The NSL+D subpattern uses only groups, whereas the SL+D pattern organizes content into FastTabs.

Wireframe

```
<Any Container>
  Toolbar
  Container Body
    Quick Filter
    Grid
    Grid Container
  Details Section
    Details Header Group
    Details Body Container
```

Pattern changes

Here are the main changes to this pattern since Microsoft Dynamics AX 2012:

- This pattern is new. Any pattern changes to the SL+D pattern can be found in the Simple List and Details pattern document.
Model

High-level structure
- <Container>
  - ActionPane (ActionPane Style=Strip)
  - ContainerBody (Group Columns=2)
    - ListContainer (Group)
      - Grid | Tree | ListView
    - DetailsContainer (Group)
      - DetailsHeader (Group)
      - DetailsGroup (Group) [Optional]

Core components
1. Apply the NestedSimpleListDetails subpattern to the container control.
2. Resolve the required BP checks:
   a. Set Grid.Datasource=<secondary data source>.
   b. Set grid data source InsertIfEmpty=No.
   c. Set ActionPane.DataSource=<same data source as grid>.
   d. Set Toolbar Command Add properties.
   e. Set Toolbar Command Remove properties.
   f. If the grid data source is read-only, make sure that there are no Add/Remove buttons on the Toolbar.

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with the UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in a browser, and walk through these steps.

Standard form guidelines
- Standard form guidelines have been consolidated into the General Form Guidelines document.

Nested simple list & detail guidelines
- There should not be duplicate New or Delete buttons.
- If a grid is used for the list portion of the pattern:
  - The grid should be a list-style grid.
  - List-style grids should display no more than three rows (lines) for each record in the List style grid. Typically, just the ID and Description are sufficient.
  - When there is no data, the grid control should not automatically add a new record.
- A details section that is displayed on the right of the Container Body:
  - Display the grid columns as the first fields in the Details Header Group, in the same order that they are displayed in the grid.
  - When a record is added, focus should go to the first field in the details section.

Examples
Form: HcmJob (TaskTabPage)
Resources

Typically used by patterns

- Simple List and Details
- Table of Contents
- Details Master
- Details Transaction

Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

- The details area of the nested pattern should not have FastTabs. The framework should verify/enforce this.
  - Currently we aren’t allowing tabs of any kind inside this pattern.

AX 2012 content
This topic provides information about the Section Chart form pattern. This pattern is primarily used in conjunction with the Operational Workspace pattern, and specifically on forms that contain a chart control.

Usage

The Section Chart form pattern is intended to be used primarily in conjunction with the Operational Workspace pattern. Specifically, the chart section or summary section contains Form Part Controls that point to forms that contain charts. These referenced forms are intended to use the Section Chart pattern.

Wireframe

Pattern changes for Finance and Operations

This pattern didn’t exist for Microsoft Dynamics AX 2012.

Model

High-level structure

- Form Design
  - HeaderGroup (Group) [Optional] – This uses one of the Filters and Toolbar subpatterns.
  - Chart

Core components

Apply the Section Chart pattern to the appropriate form/container.

Related container patterns

- Workspace
- Section stacked chart

UX guidelines
The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps. None

Examples

Form: FmBiChartPart_VehicleByModel (All workspaces > Reservation Management (see the Statistics section)

Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

None
This article provides information about the Section PowerBI subpattern. This subpattern is used as part of the Operational Workspace pattern, specifically for the panorama section that contains a PowerBI control.

Usage

The Section PowerBI subpattern is used as part of the Operational Workspace pattern, specifically for the panorama section that contains the PowerBI control.

Wireframe

Pattern changes for Microsoft Dynamics AX

This pattern didn't exist for Microsoft Dynamics AX 2012.

Model

**High-level structure**
TabPage PowerBI (PowerBI)

**Core components**
Apply Section PowerBI to the appropriate tab page in the workspace.

**Related container patterns**
- Operational workspace

UX guidelines

None
Examples

Form: FmClerkWorkspace (All workspaces > Reservation Management) PowerBI must be configured before the form can appear. (For information about how to configure PowerBI, see the Appendix.)

Appendix

Related articles

- Configure Power BI integration for workspaces
- Features and services available through Power BI integration

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

- How do I configure PowerBI for integration with my workspace?
  - See the Configure Power BI integration for workspaces article.

Open issues

None
This article provides information about the Section Related Links subpattern. This subpattern is used as part of the Operational Workspace pattern, specifically for the last panorama section that contains a set of links to other forms.

Usage

The Section Related Links subpattern is used as part of the Operational Workspace pattern, specifically for the last panorama section that contains a set of links to other forms.

Wireframe

```
Panorama TabPage
  Link (0..n)
  Group (0..n)
    Link (1..n)
```

Pattern changes for Microsoft Dynamics AX

This pattern didn't exist for Microsoft Dynamics AX 2012.

Model

- **High-level structure**
  - TabPage
    - LinkButton ($Button) [0..N]
    - ButtonGroup (Group) [0..N]
  - LinkButton ($Button) [1..N]

- **Core components**

  Apply Section Related Links to the appropriate tab page in the Operational Workspace.

- **Related container patterns**
  - Operational Workspace

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps. None
Examples

Form: PurchOrderProcessReceiptsWorkspace (All workspaces > Purchase order receipt and follow-up) (see the Links section)

Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

None
This article provides information about the Section Stacked Chart subpattern. This subpattern is used as part of the Operational Workspace pattern when a panorama section contains one or two charts.

Usage

The Section Stacked Chart subpattern is used as part of the Operational Workspace pattern, specifically for a panorama section that contains one or two charts.

Wireframe

Pattern changes for Microsoft Dynamics AX

This pattern didn't exist for Microsoft Dynamics AX 2012.

Model

High-level structure

- TabPage
  - ChartPart (FormPart) [0..N]

Each Form Part points to a form that contains a single chart. Each of these forms should use the Section Chart form pattern.

Core components

Apply Section Stacked Chart to the appropriate tab page in the workspace.

Related container patterns

- Operational workspace
UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

- There should be no more than two charts in this section.
- Each Form Part Control should point to a form that uses the Section Chart pattern.

Examples

Form: FmClerkWorkspace (All workspaces > Reservation Management)

Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues
This article provides information about the Section Tabbed List subpattern. This subpattern is used as part of the Operational Workspace pattern, specifically for a panorama section that contains a set of vertical tabs, each of which contains a filtered list of data.

Usage

The Section Tabbed List subpattern is used as part of the Operational Workspace pattern, specifically for a panorama section that contains a set of vertical tabs, each of which contains a filtered list of data.

Wireframe

```
Tabbed list section (Tab page)

Tab

TabPage (1..n)

Form Part
```

Pattern changes for Microsoft Dynamics AX

This pattern didn't exist for Microsoft Dynamics AX 2012.

Model

**High-level structure**

- TabPage
- TabbedList (Tab) (Style=VerticalTabs)
  - TabbedListPage (TabPage) [0..N]
  - TargetForm (FormPart)

Each Form Part points to a form that contains the content for the section. Each of these forms should use one of the Form Part Section List form patterns.

**Core components**
Apply Section Tabbed List to the appropriate tab page in the workspace.

**Related container patterns**
- Operational Workspace
- Form Part Section List

**UX guidelines**

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

- At least one list should be present in the tabbed list section.
- Each Form Part Control should point to a form that uses one of the Form Part Section List patterns.

**Examples**

Form: **PurchOrderMaintainWorkspace** *(All workspaces > Purchase order preparation)*

![Order form screenshot]

**Appendix**

**Frequently asked questions**

This section will have answers to frequently asked questions that are related to this guideline/pattern.

**Open issues**

None
This article provides information about the Section Tiles subpattern. This subpattern is used as part of the Operational Workspace pattern, specifically for the first panorama section (the Summary section) that contains a set of tiles, charts, and singleton cards.

Usage

The Section Tiles subpattern is used as part of the Operational Workspace pattern, specifically for the first panorama section (the Summary section) that contains a set of tiles, charts, and singleton cards.

Wireframe

![Wireframe Diagram]

Pattern changes for Microsoft Dynamics AX

This pattern didn't exist for Microsoft Dynamics AX 2012.

Model

**High-level structure**

- TabPage
  - TileButton (TileButton) [0..N]
  - TargetForm (FormPart) [0..N]

The Form Parts are used to embed Charts or singleton Cards into the Summary section of the workspace. Each form that represents a Chart should use the Section Chart form pattern.

**Core components**

Apply Section Tiles to the first tab page in the Operational Workspace.
Related container patterns

- Operational Workspace
- Section Chart

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

- The Summary section should be named “Summary” or a variant that qualifies the word “Summary.”
- No two tiles in the workspace should have the same symbol.
- There should be a maximum of one “New” tile.
- Chart sizes should correspond to multiples of tile sizes.
  - Available sizes include 1 tile tall × 2 tiles wide, 2 × 2, 2 × 3, 2 × 4, 2 × 6, 4 × 4, 4 × 6, and 4 × 8.

Examples

Form: PurchOrderMaintainWorkspace (All workspaces > Purchase order preparation) (see the Summary section)

![Summary section](image)

Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

None
This article provides information about the Tabular Fields subpattern. This subpattern is used to show information efficiently in a tabular format.

Usage
This subpattern is used to show information efficiently in a tabular format. The fields are arranged in a table that contains rows and columns, and that optionally contains column headers, row labels, a caption, and a footer. The Tabular Fields subpattern can be applied on the following controls:

- TabPage control
- Group control

Wireframes

Structural wireframe

Pattern changes
In previous releases of Microsoft Dynamics AX, there was no formally accepted way to model this pattern. Therefore, this pattern was modeled in many inconsistent ways that must be modified to match the current pattern. The most common way to model this pattern was to use groups for columns. However, groups are now used for the rows. The primary reason for this change was to better match the HTML/CSS constructs, and it also helps keep the tab sequence and semantics of a table.

Model

**High-level structure**
- TabularFields (Group*)
  - CaptionGroup (Group)
    - TableCaption (StaticText) [Optional]
  - TableHeaderRow (Group)
    - Column0Label (StaticText) [Optional] – **Note:** This static text fills col0, row0 with a blank.
    - ColumnLabels (StaticText) [1..N] – **Note:** These are the normal column headers.
  - TableRows (Group) [1..N]
    - RowLabel (StaticText) [Optional]
    - RowValues ($Field) [1..N] OR SecondaryColumnLabel (StaticText) [1..N]
  - TableFooterGroup (Group)
    - Column0Label (StaticText) [Optional] – **Note:** This static text fills col0, footer with a blank.
    - RowValues ($Field) [0..N] – **Note:** All the footer fields are in view mode.

Note that the four groups in the top-level tabular fields are mandatory structural elements. However, the contents of all those groups exception the Rows (Group) are optional. Additionally, note that Tabular Fields can also be used on a TabPage control. The structure is the same as the structure that is shown here.

**Core components**

Apply the Tabular Fields pattern on the top-level group or tab page. Address the pattern errors and problems.

**UX guidelines**

No manual verification is required.

**Examples**

Form: LedgerJournalTransVendPaym (Balances) (Accounts payable > Journals > Payment journal > Lines)

<table>
<thead>
<tr>
<th></th>
<th>DEBIT</th>
<th>CREDIT</th>
<th>DIFFERENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOUCHER</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>JOURNAL</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

**Resources**

**Typically used by patterns**
- Simple List and Details
- Table of Contents
- Details Master
Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

- Why are we changing how the tabular field layout is created?
  - To accomplish the layout in HTML, we must align with the way that HTML layout works. HTML layout groups by rows, not by columns.

Open issues

- None
This article provides information about the Toolbar and Fields subpattern. This container pattern is used to show actions above a subpattern of data fields. The toolbar should contain fewer than 10 actions.

Usage

This container pattern is used to show actions above a subpattern of data fields. The toolbar should contain fewer than 10 actions.

Wireframe

![Toolbar and Fields Wireframe](image)

Model

**High-level structure**
- [Container]
  - Toolbar (ActionPane, Style=Strip)
  - ContentGroup (Group) – **Note**: A fields subpattern is used.

**Core components**
- Apply the ToolbarFields subpattern to the container control.
- Address BP Warnings:
  - No additional BP checks are required beyond the AX6.3 BP checks that were carried forward.

**Related patterns**
- Toolbar and List

**Commonly used subpatterns**
- Fields and Field Groups
- Tabular Fields
- Dimension Expression Builder

**UX guidelines**

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development
environment. Open the form in the browser, and walk through these steps.

**Standard form guidelines:**

- Standard form guidelines have been consolidated into the [General Form Guidelines](#) document.

**Toolbar guidelines:**

- Toolbar guidelines have been consolidated into the Dynamics AX [General Form Guidelines](#) document.

### Examples

**Toolbar and Fields**

**Form:** HcmPosition (WorkerAssignmentTabPage)

![Worker assignment form](image)

**Resources**

**Typically used by patterns**

- Simple List and Details
- Table of Contents
- Details Master
- Details Transaction

### Appendix

**Frequently asked questions**

This section will have answers to frequently asked questions that are related to this guideline/pattern.

**Open issues**

- Should the ShowMoreLess group be part of the pattern, or should it be its own subpattern?
  - We will treat the `ShowMoreLess` group as a custom container pattern until there is enough demand to justify the addition of a new pattern.

**Microsoft Dynamics AX 2012 content**

**HcmPosition**

![HcmPosition form](image)
This article provides information about the Toolbar and List form subpattern. This subpattern is used to show child collections for the parent entity as either a tabular grid or a tree.

Usage

This subpattern is used to show child collections for the parent entity as either a tabular grid or a tree. The toolbar contains fewer than 10 actions. If a grid is used, it contains fewer than 10 fields. This article describes two patterns:

- **Toolbar and list** – This is the basic version of the pattern and should be used by default.
- **Toolbar and list (double)** – This variant includes two lists and an optional toolbar above each list.

Wireframes

**Toolbar and list**

<table>
<thead>
<tr>
<th>Container</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Toolbar (optional)</strong></td>
</tr>
</tbody>
</table>

| Custom Filter Group (optional) |

| Grid | Tree | ListView | Table |

| Footer (optional) |

**Toolbar and list (double)**
### Model

#### Toolbar and list – High-level structure
- [Container]
  - Toolbar (ActionPane, Style=Strip) [Optional]
  - CustomFilterGroup (Group) [Optional]
  - Grid | Tree | ListView | Table
  - Footer (Group) [Optional]

#### Toolbar and list (double) – High-level structure
- [Container]
  - Toolbar1 (ActionPane, Style=Strip) [Optional]
  - CustomFilterGroup1 (Group) [Optional]
  - Grid | Tree | ListView | Table
  - Toolbar2 (ActionPane, Style=Strip) [Optional]
  - CustomFilterGroup2 (Group) [Optional]
  - Grid | Tree | ListView | Table
  - Footer (Group) [Optional]

#### Core components
- Apply the ToolbarList subpattern to a container control.
- Address BP Warnings:
  - Grid.DataSource must not be empty.
  - No additional BP checks are required beyond the AX6.3 BP checks that were carried forward.

#### Related patterns
- Toolbar and Fields
UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn't include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

Standard form guidelines:

- Standard form guidelines have been consolidated into the Microsoft Dynamics AX General Form Guidelines document.

Toolbar guidelines:

- Toolbar guidelines have been consolidated into the Dynamics AX General Form Guidelines document.

Examples

Toolbar and list

Form: VendTable (TabCommunication)

<table>
<thead>
<tr>
<th>Contact information</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIPTION</td>
<td>TYPE</td>
</tr>
<tr>
<td>A. Datum Corporation</td>
<td>Phone</td>
</tr>
<tr>
<td>A. Datum Corporation</td>
<td>URL</td>
</tr>
</tbody>
</table>

Toolbar and list (double)

Form: SalesQuickQuote (TabPageExistingItems)
Resources

**Typically used by patterns**
- Simple List and Details
- Table of Contents
- Details Master
- Details Transaction

Appendix

**Frequently asked questions**
This section will have answers to frequently asked questions that are related to this guideline/pattern.

**Open issues**
- Does `CommandButton.Command` contain Add and Remove actions that provide the default icon, label, and tooltip?
  - Not yet. Deliverable 1052359 will look at setting these defaults for Add and Remove commands.
- Do we care about separators between groups of icons in a migration context?
  - Currently, we don't show separators between button groups on toolbars.
<table>
<thead>
<tr>
<th>Description</th>
<th>Type</th>
<th>Contact number/address</th>
<th>Extension</th>
<th>Primary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forest Wholesales Fax (One-time)</td>
<td>Fax</td>
<td>111-555-0140</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Forest Wholesales URL</td>
<td>URL</td>
<td><a href="http://www.customer40.consolidatedmess">http://www.customer40.consolidatedmess</a>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Forest Wholesales Email</td>
<td>E-mail address</td>
<td><a href="mailto:forest.wholesales@customer40.consolidated">forest.wholesales@customer40.consolidated</a>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Forest Wholesales Phone (One-time)</td>
<td>Phone</td>
<td>111-555-0100</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Forest Wholesales Fax (One-time)</td>
<td>Fax</td>
<td>321-555-0152</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This article provides information about the Workspace Page Filter Group subpattern. This subpattern is used as part of the Operational Workspace pattern when a workspace must expose a single workspace-wide filter on the form.

Usage

The Workspace Page Filter Group subpattern is used as part of the Operational Workspace pattern, specifically when a workspace must expose a single workspace-wide filter on the form.

Wireframe

![Wireframe Diagram]

Pattern changes for Microsoft Dynamics AX

This pattern didn’t exist in Microsoft Dynamics AX 2012.

Model

High-level structure

- Group
  - Field ($Field)

Core components

Apply Workspace Page Filter Group to the appropriate group in an (operational) workspace.

Related container patterns

- Custom Filter Group
- Operational workspace

UX guidelines

The verification checklist shows the steps for manually verifying that the form complies with UX guidelines. This checklist doesn’t include any guidelines that will be enforced automatically through the development environment. Open the form in the browser, and walk through these steps.

- The filter field should have a drop-down that contains the list of available values.
- The expectation is that the filter field will be modified multiple times per day by the target user. If the filter field is modified less often or is primarily static, it should be put in a configuration dialog.
- If more filter fields are needed, they should be put in a configuration dialog (up to five filter fields).
Examples

Form: ReqCreatePlanWorkspace (All workspaces > Master Planning)

Appendix

Frequently asked questions

This section will have answers to frequently asked questions that are related to this guideline/pattern.

Open issues

None
This topic describes how to create new application controls that have a property sheet in Visual Studio and have server-side business logic.

Prerequisites

For this tutorial, you must access the environment by using Remote Desktop, and you must be provisioned as an administrator on the instance. For more information, see Deploy and access development environments.

Overview

The Control Extensibility Framework lets you create new application controls. You can use the same tools that Microsoft uses to build controls that are already present in the program, such as the chart control. Three important artifacts are involved in the process of developing an extensible control:

- **The X++ build class** – The build class lets a developer define the properties that appear in the Microsoft Visual Studio property sheet for the control. The developer can also define the modeling behavior for the control when it’s used in the form designer. The build class is consumed by the run-time class to initialize the state of the control based on the value of properties in the property sheet.

- **The X++ run-time class** – The run-time class lets a developer define server-side business logic and data access patterns for an extensible control. Two concepts that are specific to building extensible controls are the properties and commands that the X++ class defines. Each property and command that is defined is serialized into a JavaScript view model at run time, and can be consumed by the client parts of the extensible control (the HTML and JavaScript). These properties and commands are the main channels for moving information between the server-side and client-side parts of the control.

- **The control HTML and JavaScript** – Each control uses HTML, JavaScript, and CSS files to define control visualization and client-side interaction patterns. By using the Microsoft Dynamics HTML binding syntax together with jQuery, a developer can consume the properties and commands that are defined in X++ to design powerful data-driven UI.

All three artifacts of extensible control development are explained in more detail in the following sections.

Key concepts

- Defining an extensible control’s design-time behavior
- Defining an extensible control’s run-time behavior
- Defining an extensible control’s view by using HTML and CSS
- Defining an extensible control’s view model by using JavaScript

Setup

**Import the tutorial project and transactional data**

Use Visual Studio to import the tutorial project. The tutorial project includes the artifacts that you will use to complete this tutorial. Use Visual Studio to open the FMTutorial project and load the data for the tutorial. You will use the FMTDataHelper class to load data for the Fleet Management tutorial.

1. Download the Fleet Management sample from https://github.com/Microsoft/FMLab, save it to C:\, and unzip it.
2. On the desktop, double-click the Visual Studio shortcut to open the development environment.

3. On the Dynamics 365 menu, click **Import Project**.

4. In the Import Project dialog box, next to the File name text box, click the ellipsis button (...).

5. In the Select the file to import dialog box, browse to C:\FMLab, click FMTutorialDataModel.axpp, and then click Open.

6. In the Project file location field, enter C:\FMLab.

7. Select the Overwrite Elements check box and the Current solution option. The following screen shot shows the completed Import Project dialog box.

8. Click OK.

9. In Solution Explorer, under the FMTutorial project, expand Classes.

10. Right-click FMTDataHelper, and then click Set as Startup Object.

11. On the BUILD menu, click Rebuild Solution. You use the rebuild to make sure that all the files in the project are built, regardless of timestamps. You can view the build progress in the Output window.

12. After the build is completed, press Ctrl+F5 to run the project. The Login form closes when authentication succeeds, and then the data is loaded.

**Set up aggregate data**

Use FMTAggregateMeasurements to populate the Microsoft SQL Server Analysis Services database with aggregate data.

**NOTE**

These steps must be completed immediately after you use the FMTDataHelper class to import data. You may NOT need to do these steps if the aggregate measure is "InMemoryRealTime", depending on what tutorial files you have.

1. In Solution Explorer, under Analytics, double-click FMTAggregateMeasurement.

2. In the designer, right-click FMTAggregateMeasurement, and then click Deploy and Process.
Preview the clerk workspace

Before you begin to build the contact control, look at the appearance of the current implementation. In the following sections, you will use the Control Extensibility Framework to enrich the visualization of the controls and the form.

1. In Solution Explorer, expand Forms, right-click FMTClerkWorkspace, and then click Set as Startup Object.

2. Press Ctrl+F5 to open the Fleet management clerk page in Internet Explorer. As the following screen shot shows, the data on this page appears as a simple grid in a list style that contains several string and date controls.

3. Exit Internet Explorer.

Modify the build class for the contact control

To save time, you will work on a partially completed extensible control that is named the contact control. You will extend the contact control to complete its design, run-time, and visualization behaviors. The partially completed contact control already supports multiple title fields, subfields, and action buttons. However, it doesn’t currently support an image. To add image support, you must extend the design experience for the contact control. You will add a data field that can specify image data.

Technical overview

To see an example of a build class, in Solution Explorer, expand Classes, right-click FMTBuildContactControl, and then click View Code. The class code appears in the code editor. FMTBuildContactControl is the build class for the contact control. For each extensible control, the build class defines the properties that the control shows in the property sheet. The build class also defines the modeling experience for the control in the Visual Studio form designer. There are three primary design-time behaviors that you can define for an extensible control. Each behavior is declaratively defined by using a FormDesign attribute. Here are the design-time behaviors that you can define:

- **Name** – You can specify the control name that appears in the form designer when you add the control to a form. To specify the name, add a FormDesignControlAttribute attribute to the build class declaration of the extensible control. For example, the following declaration of the FMTBuildContactControl class shows the attribute.
**Designer properties** – These are the properties that you see in the property sheet when you add the control to a form. There are several attributes that let you add various types of designer properties. For example, the `FormDesignPropertyAttribute` attribute adds a property to the property sheet, and the property name and the section are supplied as arguments to the attribute. For example, the following code adds the `Action Name` property to the `FMTContactControlAction` class.

```
[FormDesignPropertyAttribute("Action Name", "Data Binding")]
public str parmActionName (str _actionName = actionName){...}
```

The following screen shot shows how this property appears in the **Properties** pane in Visual Studio.

**Child design components** – These are child nodes that you see after you add the control to a form. There are two types of child design components: leaf and leaf collection.

- A leaf is defined by using a `FormDesignComponentAttribute` attribute on an X++ method that accepts or returns another build class. The build class determines the properties that the leaf has in the property sheet.
- A leaf collection is defined by using a `FormDesignComponentCollectionAttribute` attribute. The allowable leaf types for the collection are defined by using `FormDesignComponentValidChildAttribute` attributes.

For example, the following code adds a leaf collection that is named `Actions` for the `FMTBuildContactControl` class.

```
[FormDesignComponentCollectionAttribute("Actions"),
 FormDesignComponentValidChildAttribute(classstr(FMTContactControlAction), "Action")]
public List parmActions(List _actions = actions){...}
```

The following screen shot shows how the specified child design component appears when you add the control to a form.
Tutorial steps

1. Check that the code for the FMTBuildContactControl class appears in the code editor. If it doesn't, in Solution Explorer, expand Classes, right-click FMTBuildContactControl, and then click View Code.

2. Add a child design component to the FMTBuildContactControl class. A child design component lets a developer who places the control in a form to specify the image that appears on the control. In this step, you will add the FormDesignComponentAttribute attribute to create a new entry in the property sheet. You will then add the FormDesignPropertyDataFieldAttribute attribute, which indicates that the new designer property enables the selection of a data field.

   a. Add the highlighted code that follows to the declarations for the class. This code adds the FormBindingDataField field to the X++ that the FMTBuildContactControl class is using.

   ```
   [FormDesignControlAttribute("FMT Contact Control")]  
class FMTBuildContactControl extends FormBuildControl  
{  

str dataSource;  
FormBindingDataField imagefield;  
List titleFields;  
List subFields;  
List actions;  
}
   ```

   b. Add the following code to the FMTBuildContactControl class. Add this method after the designer property for the data source.

   ```
   [FormDesignComponentAttribute("Image")]  
public FormBindingDataField parmImageField(FormBindingDataField _imageField = imageField)  
{  

if(!_prmisdefault(_imageField))  
{

_imageField = _imageField;

}

return imageField;
}
   ```

   NOTE

   The child design component will show the properties that are available on the FormBindingDataField build class. This is appropriate, because you want to enable image data binding to a data field and data source. This is all that is required to add a designer property to the build class of the contact control.

3. Press Ctrl+S to save your changes, and then close the code editor.

4. In Solution Explorer, right-click FMTutorial, and then click Build.

5. If the FMTPickingUpTodayPart form isn't already open, expand Forms, and then double-click
Modify the runtime class for the contact control

Next, you must modify the runtime class to read the data source and data field for the image from the build class. You must also create a runtime property, so that the image data is available to the control's client HTML and JavaScript.

Technical overview

To see an example of the runtime class, in Solution Explorer, expand Classes, right-click FMTContactControl, and then click View Code. The class opens in the code editor. FMTContactControl is the runtime class for the contact control. The class defines the runtime behavior of the contact control. The runtime class typically contains X++ for data access or business logic. In addition, there are two primary runtime behaviors that are related to extensible controls that you define in the runtime class. Each behavior is declaratively defined by using an attribute.

- Run-time properties of the control – These properties can be of two types:
  - Static properties, which are set via code or initialized with values from designer properties.
  - Bindable properties, for which the runtime value is determined by a binding to a data source and data field combination.

Run-time properties are declared by using FormPropertyAttribute attributes. The following example shows a property declaration in FMTContactControl.

```csharp
[FormPropertyAttribute(FormPropertyKind::Value, "TitleFields")]
public List parmTitleFields(List _value = titleFieldsProperty.parmValue()){}
```

The FormPropertyAttribute attribute accepts two arguments:

- The first argument indicates to the framework the kind of JavaScript view model property to create.
  - If you supply BindableValue, a ReferenceProperty is generated in the JavaScript view model. A ReferenceProperty updates itself when data changes in the data source.
  - If you supply Value, a ValueProperty is generated in the JavaScript view model. A developer
must write code to update the value of a **ValueProperty**.

— The second argument of the attribute sets the name for the property as it will be defined in the JavaScript view model.

**NOTE**

Don’t be concerned if **TitleFields** don’t seem to be bound to data because the example uses a **Value** property. The **TitleFields** property returns a List that contains **FormBindingDataFields**, each of which is data-bound. The X++ method that has the **FormPropertyAttribute** attribute is a simple getter/setter that uses a **FormProperty** as the backing field. The **FormProperty** contains the logic for updating the property, based on value or data source changes. It also serves as the backing field for the property.

---

**Run-time commands for the control** – Commands enable the client parts of the control to trigger X++ logic, based on client-side user interactions. Commands are declared by using a **FormCommandAttribute** attribute. The single argument specifies the name of the command as it will appear in the JavaScript view model. The following example shows a command declaration in **FMTContactControl**.

```x++
[FormCommandAttribute("ExecuteAction")]
public void executeAction(str _actionName){...}
```

---

**Tutorial steps**

1. Verify that the **FMTContactControl** class is open in the code editor. If it isn’t, in Solution Explorer, expand **Classes**, right-click **FMTContactControl**, and then click **View Code**.

2. Add a run-time property for the image data to **FMTContactControl**. In the **FMTContactControl** class, declare a **FormProperty** that is named **imageFieldProperty**, as shown by the highlighted line in the following example.

```x++
[FormControlAttribute(...)]
class FMTContactControl extends FormTemplateControl {
    FormProperty dataSourceProperty;
    FormProperty imageFieldProperty;
    FormProperty titleFieldsProperty;
    FormProperty subtitleFieldsProperty;
    FormProperty actionsProperty;
}
```

3. Add the following X++ method after the **parmDataSource** X++ method. The new method will serve as the getter/setter for **imageFieldProperty**.

**NOTE**

You don't return the value of the image data here, because the framework will let you bind to the data in the client, as you will see later.
4. Initialize `imageFieldProperty` by adding the highlighted line in the following example to the new method of `FMTContactControl`.

```java
public void new(FormBuildControl _build, FormRun _formRun)
{
    super(_build, _formRun);

    this.setTemplateId("FMTContactControl");
    this.setResourceBundleName("/Resources/Html/FMTContactControl");

    dataSourceProperty = this.addProperty(methodstr(FMTContactControl, parmDataSource), Types::String);
    imageFieldProperty = this.addProperty(methodstr(FMTContactControl, parmImageField), Types::String);
    titleFieldsProperty = this.addProperty(methodstr(FMTContactControl, parmTitleFields), Types::Class);
    subtitleFieldsProperty = this.addProperty(methodstr(FMTContactControl, parmSubtitleFields), Types::Class);
    actionsProperty = this.addProperty(methodstr(FMTContactControl, parmActions), Types::Class);
}
```

5. Now supply the binding to `imageFieldProperty` by adding the highlighted line in the following example to the `applyBuild` method of `FMTContactControl`.

```java
if(build)
{
    this.parmDataSource(build.parmDataSource());
    this.parmImageField(FormBindingUtil::initBinding(build.parmImageField(), parmDataSource(), build.parmImageField(), parmDataField(), this.formRun()));
}
```

6. Press Ctrl+S to save the changes. You’ve now finished modifying the run-time class. Next, you will update the HTML view to display the image.

**Modify the HTML for the contact control**

The HTML of the contact control is where you add UI elements, such as text boxes, images, and buttons, that interact with the properties and commands that are defined in the run-time class. Extensible controls use a declarative HTML-based binding syntax to bind HTML element behaviors to properties, commands, JavaScript expressions, and JavaScript functions. These bindings are parsed at run time, and the resulting HTML is injected.
into the DOM. The following section explains a few of the bindings that are used in FMTContactControl.htm to add an image to the control.

**Technical overview**

The **bind** attribute, together with the **text** binding handler enables binding to the **text** property of an HTML element. For example, the following HTML uses the **bind** attribute and the **text** binding handler.

```html
<span data-dyn-bind="text:'Hello World!'"/>
```

The preceding HTML is equivalent to the following HTML.

```html
<span>Hello World!</span>
```

You will see the benefits of the binding when you bind to properties or commands. For example, if you have a view model property that is named **FirstName**, you can bind to it as shown in the following example. Here, **$data** is the object that contains the view model properties and commands.

```html
<span data-dyn-bind="text: $data.FirstName"></span>
```

The HTML output changes, based on the current value of **FirstName**. The following example shows the output if **FirstName** has a value of **John**.

```html
<span>John</span>
```

If the value of the **FirstName** property changes for some reason (for example, X++ or JavaScript was run to update the property), the binding is automatically reevaluated, and the HTML output immediately reflects the change. All binding handlers follow this pattern of automatic reevaluation when the binding value changes. The **if** and **foreach** binding handlers are unique in that they perform DOM manipulation based on the binding values.

- To conditionally add an element to the DOM, use the **if** binding handler and supply the condition under which the element should be added. If the condition is false, the element isn't added to or removed from the DOM, and no bindings that are associated with the element are evaluated. Of course, if the binding value that is supplied to **if** changes, an element that was removed will be added to the DOM again, and the bindings will be evaluated.

- To iterate over an array of elements, use the **foreach** binding. This is useful when nearly identical HTML elements must be displayed.

The following table shows some of the other binding handlers.

<table>
<thead>
<tr>
<th>BINDING HANDLER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>css</td>
<td>Specify a CSS class, based on a condition.</td>
</tr>
</tbody>
</table>
### Style

Apply CSS styles, and bind the values to properties.

### Attr

Bind an HTML attribute.

In addition to using HTML elements inside the HTML for your control, you can also add framework controls such as CheckBox, Group, Tile, SectionContainer, Label, and List to your control. Instead of binding handlers, each framework control enables binding values to be passed to its view model properties. For example, a CommandButton is added by using the role attribute.

```html
<div data-dyn-role="CommandButton"></div>
```

In this case, ActionCommand can be supplied with a JavaScript function.

```html
<div data-dyn-role="CommandButton"
     data-dyn-bind="ActionCommand: function(){alert('Hello World!')}"></div>
```

One additional feature of the HTML binding syntax is the context-aware nature of bindings. By default, the context of all HTML elements is set to the JavaScript view model for the control. However, the context changes in certain circumstances. For example, for a foreach binding, every child element that is nested inside the hosting element (the element that has the foreach binding) obtains the current item in the loop as the context. To access the context of the parent element when you're inside of a foreach binding, use the $parent object. The following example from FTMContactControl.htm will help make this point clearer.

```html
<div data-dyn-bind="foreach: Actions">
    <div data-dyn-role="CommandButton"
         data-dyn-bind="ImageType: 'Symbol', ImageName: $data.ActionName,
                         ActionCommand: function(){$parent().ButtonClicked($data)}"></div>
</div>
```

Actions is a List property that is available on the control's JavaScript view model. This property was defined in the FMTContactControl run-time class. Each action in the Actions list has Data Source, Data Field, and Action Name properties. Within the foreach loop, $data refers to the current action, and $data.ActionName cam retrieve the ActionName property from the current action in the loop. Within the loop, view model properties on the control aren't accessible via $data. Instead, $parent can be used to retrieve the view model properties.

### Tutorial steps

Add the HTML for the ImageField property that you created in the run-time class.

1. In Solution Explorer, expand the Resources folder under the FMTutorial project, and double-click FMTContactControlHTM. The FMTContactControl.htm file opens in the HTML editor.

2. Add the following HTML to the FMTContactControl.htm HTML. The gray text is shown just for placement context.
3. Press Ctrl+S to save the changes to FMTContactControl.htm.

In the preceding example, you use the framework image control to render the image. **Value** is a property that is defined on the Image control. This property lets you specify the value for the image data. The image control supports several kinds of image types, but for this example, you’re concerned with only two possible types: URLs and Base64 strings. Because the image type depends on data that is known only at run time, you will use a property that derives this information, **ImageValue**. You might notice that no such property is defined in the run-time class for **FMTContactControl**. Therefore, this property isn’t part of the automatically generated JavaScript view model for that control, and it also isn’t defined on $data. To make the **ImageValue** property accessible via $data, you must extend the automatically generated JavaScript view model to add the property.

**Review the JavaScript for the contact control**

As was mentioned earlier, for every X++ method that has either a **FormPropertyAttribute** or **FormCommandAttribute** attribute, a JavaScript property or command is generated and made accessible to an extensible control’s HTML via the view model. You can extend this view model with additional properties and commands that are defined only on the client. In other words, the properties and commands have no associated X++ methods. After you extend the view model, the additional client-only properties and commands can be used in bindings via the $data object.

**Technical overview**

The Control Extensibility Framework offers many functions that help with data bindings and data access. Some of the functions that are used in FMTContactControl.htm, such as **$field** or **$model**, make it easy to access the data source and its fields from the HTML bindings. These functions are functional aliases that are used in the HTML bindings for JavaScript functions that are defined by the framework. Within the extended JavaScript view model, the equivalent, non-aliased functions are **$dyn.getField** and **$dyn.getModel**. You can also use jQuery within the extended JavaScript view model by using the $ symbol. The following example shows the standard pattern that is used to define a constructor for the extended JavaScript view model. In this example, you save a reference to **this**, apply the base **Control** class behaviors, and then combine the automatically generated properties and commands with the properties and command from the extended view model.

```javascript
$dyn.controls.ContactControl = function (props) {
    var self = this;

    $dyn.ui.Control.apply(this, arguments);

    $dyn.ui.applyDefaults(this, props, $dyn.ui.defaults>ContactControl);

    ...
}
```

The **self** variable now contains all properties and commands that are generated from the X++ run-time class. The following example shows how to add a client-only property to extend the view model.
Add the extensible control to the Fleet Management workspace

You will now update the Fleet Management Clerk workspace so that it uses the contact control that you just completed.

1. In Solution Explorer, expand Forms, and then double-click FMTPickingUpTodayPart. The form opens in the form designer.

2. In the form designer, expand Design > PickingUpTodayGrid.

3. If there is an existing contact control, delete it. You must remove and then re-add the control, so that the form designer picks up the X++ changes that you made. Right-click the existing control, and then click Delete.

4. Right-click PickingUpTodayGrid, point to New, and then click FMT Contact Control.

5. Click the FMTContactControl1 node that you just added, and set the Data Source property to FMTCustomer.

6. Expand the FMTContactControl1 node, click Image, and then, in the Properties pane, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source</td>
<td>FMTCustomer</td>
</tr>
<tr>
<td>Data Field</td>
<td>Image</td>
</tr>
</tbody>
</table>

7. Create new title fields:

   a. Right-click Title Fields, and then click New Title Field.

   b. Click the Title Field node that you just created, and then, in the Properties pane, set the following properties.
<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>FirstName</td>
</tr>
<tr>
<td>Data Source</td>
<td>FMTCustomer</td>
</tr>
<tr>
<td>Data Field</td>
<td>FirstName</td>
</tr>
</tbody>
</table>

- **c.** Right-click **Title Fields** again, and then click **New Title Field**.

- **d.** Click the **Title Field** node that you just created, and then, in the **Properties** pane, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>LastName</td>
</tr>
<tr>
<td>Data Source</td>
<td>FMTCustomer</td>
</tr>
<tr>
<td>Data Field</td>
<td>LastName</td>
</tr>
</tbody>
</table>

8. Create new subtitle fields:

- **a.** Right-click **Subtitle Fields**, and then click **New Subtitle Field**.

- **b.** Click the **Subtitle Field** node that you just created, and then, in the **Properties** pane, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>StartDate</td>
</tr>
<tr>
<td>Data Source</td>
<td>FMTRental</td>
</tr>
<tr>
<td>Data Field</td>
<td>StartDate</td>
</tr>
<tr>
<td>Formatting Expression</td>
<td>Pickup (0)</td>
</tr>
</tbody>
</table>

- **c.** Right-click **Subtitle Fields** again, and then click **New Subtitle Field**.

- **d.** Click the **Subtitle Field** node that you just created, and then, in the **Properties** pane, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>EndDate</td>
</tr>
<tr>
<td>Data Source</td>
<td>FMTRental</td>
</tr>
<tr>
<td>Data Field</td>
<td>EndDate</td>
</tr>
<tr>
<td>Formatting Expression</td>
<td>Return (0)</td>
</tr>
</tbody>
</table>

- **e.** Right-click **Subtitle Fields** again, and then click **New Subtitle Field**.
f. Click the **Subtitle Field** node that you just created, and then, in the **Properties** pane, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>VehicleDescription</td>
</tr>
<tr>
<td>Data Source</td>
<td>FMTVehicle</td>
</tr>
<tr>
<td>Data Field</td>
<td>Description</td>
</tr>
</tbody>
</table>

g. Press Ctrl+S to save your changes.

9. Copy **PickingUpTodayGrid** by right-clicking in the grid and clicking **Copy**.

10. In Solution Explorer, click **Forms > FMTReturningTodayPart**, and then double-click **FMTReturningTodayPart**. The form opens in the form designer.

11. Expand **Design**, right-click **ReturningTodayGrid**, and then click **Delete**.

12. Right-click **Design**, and then click **Paste**.

13. Select the **PickingUpTodayGrid** grid that you just added to the **FMTReturningTodayPart** form. Set the **Name** property to **ReturningTodayGrid**, and then press Ctrl+S to save the changes to the **EMTReturningTodayPart** form.

14. In Solution Explorer, find the **FMTRentalRatesPart** form. Double-click the form to open it in the form designer, and then click **Design > RentalRatesGrid**.

15. Delete each field from **RentalRatesGrid**. To remove the fields, click the first field, hold down the Shift key while you click the last field, and then press Delete.

16. Right-click in the grid, point to **New**, and then click **FMT Contact Control**.

17. Expand **FMTContactControl1**, click **Image**, and then, in the **Properties** pane, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source</td>
<td>FMTVehicleModel</td>
</tr>
<tr>
<td>Data Field</td>
<td>Image</td>
</tr>
</tbody>
</table>

18. Right-click **Title Fields**, and then click **New Title Field**.

19. Click the title field node that you just created, and then, in the **Properties** pane, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>VehicleModel</td>
</tr>
<tr>
<td>Data Source</td>
<td>FMTVehicleModel</td>
</tr>
<tr>
<td>Data Field</td>
<td>Model</td>
</tr>
</tbody>
</table>

20. Right-click **Subtitle Fields**, and then click **New Subtitle Field**.
21. Click the **Subtitle Field** node that you just created, and then, in the **Properties** pane, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>VehicleMake</td>
</tr>
<tr>
<td>Data Source</td>
<td>FMTVehicleMake</td>
</tr>
<tr>
<td>Data Field</td>
<td>Make</td>
</tr>
</tbody>
</table>

22. Right-click **Subtitle Fields**, and then click **New Subtitle Field**.

23. Click the **Subtitle Field** node that you just created, and then, in the **Properties** pane, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>RatePerDay</td>
</tr>
<tr>
<td>Data Source</td>
<td>FMTModelRate</td>
</tr>
<tr>
<td>Data Field</td>
<td>RaterPerDay</td>
</tr>
</tbody>
</table>
|            | Note: The **Data Field** value must match the table field name. If you correct the spelling error, the values won't match, and you will receive a run-time error.
| Formatting Expression | $(0) per day |

24. Right-click **Subtitle Fields**, and then click **New Subtitle Field**.

25. Click the **Subtitle Field** node that you just created, and then, in the **Properties** pane, set the following properties.

<table>
<thead>
<tr>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>RatePerWeek</td>
</tr>
<tr>
<td>Data Source</td>
<td>FMTModelRate</td>
</tr>
<tr>
<td>Data Field</td>
<td>RatePerWeek</td>
</tr>
<tr>
<td>Formatting Expression</td>
<td>$(0) per week</td>
</tr>
</tbody>
</table>

26. Press Ctrl+S to save your changes to **FMTRentalRatesPart**.

27. In Solution Explorer, right-click the **FMTClerkWorkspace** form, and then click **Set as Startup Object**.

28. Press Ctrl+F5 to open the updated contact control in Internet Explorer.
Bidirectional or right-to-left support

To validate right-to-left (RTL) support for your extensible control, you simply need to set the dir (direction) attribute on the HTML document. When this attribute is changed, the browser will automatically change the layout direction of your control. You should make sure that your control doesn’t implement any styling which interferes with this layout. Instead of setting this attribute manually, you can also validate by placing your control on a form, and then selecting a RTL language. Selecting a RTL language will cause the client to also update the dir attribute appropriately. For more information, see dir attribute in the HTML standards.
Keyboard shortcuts are an important consideration when you create any extensible control. This topic provides information that will help you choose keyboard shortcuts for your extensible controls. It also outlines the recommended method for implementing keyboard shortcuts for extensible controls.

Overview

For accessibility, it's essential that keyboard-only users be able to use controls. Therefore, keyboard shortcuts are an important consideration when you create any extensible control. This topic provides information that will help you choose key combinations to use as keyboard shortcuts. It highlights the shortcuts that are currently used by Finance and Operations apps and supported browsers, shortcuts that are planned for implementation, and shortcuts that one or more browsers don't allow to be overridden. This topic also outlines the recommended way to implement keyboard shortcuts for extensible controls.

Choosing a key combination

When you're trying to choose a key combination to use as a keyboard shortcut, it's important that you be aware of other existing shortcuts. In this way, you help guarantee that your shortcut won't overlap an existing shortcut. If you try to collide with an existing shortcut, one of the following outcomes might occur:

- The new keyboard shortcut might not work, because a browser doesn't allow that key combination to be overridden, or a framework-provided shortcut takes precedence over the new shortcut.
- The new keyboard shortcut might remove expected keyboard functionality, because users expect specific key combinations to perform specific functions in a browser. Alternatively, you might override framework-provided shortcuts or other control shortcuts, so that keyboard-only users can't use them.

Because of these potential issues, we recommend that you adhere to this guidance when you choose a key combination:

- **Don't** choose any key combination that is currently used by Finance and Operations apps, or that is planned for future implementation.
- **Do** pick key combinations that will work in all supported browsers.
- **Do** be careful when you override shortcuts that are used by a supported browser. You should not suppress shortcuts for important or frequently used browser functionality.
- **Do** use longer key combinations (three keys) for control-specific behavior. Shorter combinations should be reserved for user-defined keyboard shortcuts.
- **Don't** choose any key combination that involves Ctrl+Alt, because this combination maps to Alt+Gr for some Eastern European languages and will conflict with other shortcuts.

Keyboard shortcut links

Here are links to the keyboard shortcuts that are documented for Finance and Operations apps and supported browsers:

- Keyboard shortcuts
- Microsoft Edge
- Google Chrome
- Internet Explorer 11
- Apple Safari
**Planned keyboard shortcuts**

In addition to the keyboard shortcuts that are currently used, there are several shortcuts that are planned for future implementation. To avoid conflicts with framework-provided shortcuts, you should not choose the following key combinations for extensible controls.

<table>
<thead>
<tr>
<th>SHORTCUT</th>
<th>FUNCTIONALITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>F3</td>
<td>Move to the nearest QuickFilter.</td>
</tr>
<tr>
<td>Alt+F3</td>
<td>Add a filter that is based on the value of the current control (Filter by value).</td>
</tr>
<tr>
<td>Alt+Shift+F3</td>
<td>Clear all user-defined filters.</td>
</tr>
<tr>
<td>F6</td>
<td>Move to the nearest toolbar.</td>
</tr>
<tr>
<td>Shift+F7</td>
<td>Move to the toast message.</td>
</tr>
</tbody>
</table>

**Browser/operating system keyboard shortcuts to avoid**

*Keyboard shortcuts that correspond to important functionality*

The following table provides a short, non-exhaustive list of keyboard shortcuts that correspond to important functionality in a browser or operating system. You should not choose the key combinations in this table.

<table>
<thead>
<tr>
<th>SHORTCUT</th>
<th>FUNCTIONALITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl+A</td>
<td>Select all the text in the current field, or select all content on the page.</td>
</tr>
<tr>
<td>Ctrl+C</td>
<td>Copy.</td>
</tr>
<tr>
<td>Ctrl+V</td>
<td>Paste.</td>
</tr>
<tr>
<td>Ctrl+X</td>
<td>Cut.</td>
</tr>
<tr>
<td>F5</td>
<td>Refresh the page.</td>
</tr>
<tr>
<td>Ctrl+F5</td>
<td>Refresh the page, and ignore cached content.</td>
</tr>
<tr>
<td>Shift+F10</td>
<td>Simulate a right-click.</td>
</tr>
<tr>
<td>Tab / Shift+Tab</td>
<td>Move to the next/previous control.</td>
</tr>
<tr>
<td>Ctrl+Tab / Ctrl+Shift+Tab</td>
<td>Move to the next/previous browser tab.</td>
</tr>
<tr>
<td>Alt+Tab / Alt+Shift+Tab</td>
<td>Move to the next/previous application.</td>
</tr>
<tr>
<td>Alt+Right arrow / Alt + Left arrow</td>
<td>Go to the next/previous page in the browser history.</td>
</tr>
</tbody>
</table>

**Keyboard shortcuts that can’t be overridden by some browsers**

Some browsers don’t allow the following keyboard shortcuts to be overridden. Therefore, you should not choose the following key combinations, because the shortcut won’t work in all browsers.

<table>
<thead>
<tr>
<th>SHORTCUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alt+A</td>
</tr>
<tr>
<td>Alt+T</td>
</tr>
<tr>
<td>Ctrl+F4</td>
</tr>
<tr>
<td>Alt+Tab</td>
</tr>
<tr>
<td>Alt+C</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>Alt+D</td>
</tr>
<tr>
<td>Alt+Shift+D</td>
</tr>
<tr>
<td>Alt+E</td>
</tr>
<tr>
<td>Alt+F</td>
</tr>
<tr>
<td>Alt+H</td>
</tr>
<tr>
<td>Ctrl+N</td>
</tr>
<tr>
<td>Ctrl+Shift+N</td>
</tr>
<tr>
<td>Ctrl+Shift+Q</td>
</tr>
<tr>
<td>Ctrl+T</td>
</tr>
</tbody>
</table>

### Implementing keyboard shortcuts

We recommend that you use the registration mechanism that is described in this section to implement keyboard shortcuts. There are several benefits to registering a control's shortcuts in this manner:

- You specify only the modifier keys that you want. If any other modifiers are pressed, the shortcut won't run. Therefore, a keyboard shortcut that is defined for Ctrl+Down arrow won't be triggered if Ctrl+Shift+Down arrow is pressed.
- You can reuse code.
- If you return `false` from the handler function, propagation won't stop on the event. If you return `true` (or if you don't specify a return value), propagation will stop.
- Ctrl and Meta modifiers are treated the same. Therefore, you can specify keyboard shortcuts that will work for both iOS and Microsoft Windows. For example, the shortcut Ctrl+G on Windows will be translated to Meta+G on iOS.
- Built-in telemetry for the keyboard shortcuts is used.

### Define a keyboard shortcut

1. Define a `Shortcuts` object on the control's prototype. Then define the keyboard shortcuts inside the `Shortcuts` object. These shortcuts should have the following structure. Make sure that the key code that you're trying to use is defined in the `$dyn.ui.KeyCodes` object.

```javascript
Shortcuts: {
  Name: {
    Keys: { modifier1: true, modifier2: true, keyCode: $dyn.ui.KeyCodes.* },
    // Only specify the modifiers you need
    // (between alt, ctrl/meta, shift)
    Handler: function (evt) {
      // Code to handle shortcut.
    },
    // Additional code.
  }
}
```

If more than one key code should apply to your keyboard shortcut, pass in an array of codes, as shown
Examples

Here is a Form example.

Shortcuts:

```
Save: {
  Handler: function (evt) {
    var control = evt ? $dyn.context(evt.target) : undefined;
    this.executeShortcuts(true, "SaveRecord", control);
  },
},
New: {
  Handler: function (evt) {
    var control = evt ? $dyn.context(evt.target) : undefined;
    this.executeShortcuts(true, "NewRecord", control);
  },
},
Delete: {
  Keys: {alt: true, keyCode: $dyn.ui.KeyCode.deleteKey },
  Handler: function (evt) {
    this.executeShortcuts(false, "DeleteRecord");
  },
  // Additional code
}
```

Here is a Dialogs example that extends the Form control.
Shortcuts: $dyn.extendPrototype($dyn.controls.Form.prototype.Shortcuts, {
    InvokeDefaultButton: {
        Keys: { keyCode: $dyn.ui.KeyCodes.enter },
        Handler: function (evt) {
            this.executeShortcuts(false, "InvokeDefaultButton");
        }
    },
});
This topic provides reference content for extensible control programming.

This document describes the API, HTML, and JavaScript support for creating extensible controls.

**Examples**

This document contains small code snippets that show how to use each API that is documented. More complete examples of finished controls that leverage many of these APIs can be found on Github. [Extensible Control Examples on Github]

**Control block diagram**

This high-level diagram illustrates the key components of an extensible control and how they interact with each other. Your extensible control solution will contain two X++ classes that implement your control. The runtime class implements the runtime data, presentation, and behavior of your control. The build class defines how your control is displayed in Form Designer, Property Window, and Application Explorer.

![Control block diagram](image)

**X++**

The X++ API of your control is the Form-developer-facing API. Be sure to consider the APIs and behaviors you want to provide to the Form developer when designing the X++ APIs for your control.

**Runtime: The X++ runtime class**

The runtime class defines the public, developer-facing API for your control. It also contains the runtime logic for your control. The job of the runtime class is to maintain the state of the control via the control’s properties.

**Runtime: Class declaration**

Declare an X++ class that extends [FormTemplateControl](#) or a type derived from [FormTemplateControl](#). **[FormTemplateControl](#) contains basic properties that are necessary for every control, such as the Template ID and the Resource Bundle. The following example extends the base control class, [FormTemplateControl](#).**
class MyControl extends FormTemplateControl

### Runtime: FormControlAttribute

You must apply the **FormControlAttribute** attribute to the X++ class declaration.

```x++
[FormControlAttribute(<Template ID>, <Resource Bundle Path>, <Build class name>)]
```

You must supply the following arguments:

- **Template ID**: A string that specifies the ID of the template. The Template ID is the JavaScript class name & the HTML element ID of the control. By convention, the Template ID matches the class name of the control's runtime class.

- **Resource Bundle Path**: A string that specifies the path to the resource bundle. The Resource Bundle Path is the web path where the main HTM files are located. At runtime, the Client framework will load the HTM file specified by the Resource Bundle Path. At runtime, the Client framework will search the Resource Bundle for the HTML element with an ID matching the specified Template ID. At runtime, the Client framework will instantiate the JavaScript class specified by the Template ID. This path is relative to the root of the web directory.

- **Build class name**: A string that specifies the name of the build class. The build class name is the X++ class that determines the design-time behavior. At design time, the Visual Studio framework reflects over this attribute and loads the specified X++ class as the designer class. At runtime, the X++ framework will instantiate the specified X++ class as the build class in the super of applyBuild().

The following example shows a typical class and attribute declaration for a control named "MyControl".

```x++
[FormControlAttribute('MyControl', '/resources/html/MyControl', classStr(MyControlBuild))]
class MyControl extends FormTemplateControl
```

### Runtime: FormCommandAttribute

The **FormCommandAttribute** is applied to a method in your control class, which allows the method to be called from a control's JavaScript class. A method with this attribute applied is called a **command**. Use the **FormCommandAttribute** on only the X++ methods that need to be accessed directly from the control's JavaScript class. An X++ method serving as a command can only accept string arguments. The method must perform the necessary operations to serialize or deserialize the string arguments into other types. The **FormCommandAttribute** has no effect on the behavior of the X++ method when the method is used from within X++. The **FormCommandAttribute** exposes the X++ method as an external endpoint that is accessible from JavaScript. As such, every command should be threat modeled and tested for exploits, and should perform validation on all of its arguments. The underlying X++ method should be declared private so that it is not accessible from X++. If X++ code needs to access this method's behavior, then a separate X++ method should be declared as public without the **FormCommandAttribute**. This public method should contain any shared code that is needed by both X++ and JavaScript. The private X++ method with the **FormCommandAttribute** can then call this public method to access the shared code. This practice allows the command to perform logic that is specific to calls coming from JavaScript (such as argument type deserialization, argument validation, security validation, etc.) before executing the core shared X++ logic. You supply the following arguments to the **FormCommandAttribute** constructor:

- **Name**: A required string that specifies the name of the command. A few best practices for naming Properties:
  - Capitalize the first letter, and use PascalCase.
- Use a verb in the name.
- Include the Property name if the Command is used to read/write a FormProperty (ex: `Set_<PropertyName>`)
- Do not use any of the names of inherited JavaScript properties.

- **Execute immediate**: An optional boolean that specifies whether calls to this command are deferred or require immediate execution. By default, commands have **Execute immediate** set to **true**, so calls are not deferred. Most commands likely require immediate execution because their X++ logic must run before allowing the user to complete their next action. However, commands that have no side-effects on the user’s ability to take their next action can likely be safely deferred to gain a performance benefit. For commands that can be deferred, set **Execute immediate** to **false** to reduce network chattiness.

The following example declares a command with the name of “SetText”.

```csharp
[FormCommandAttribute("SetText")]
private void setText(str value)
{
    // Add implementation code here.
}
```

**Runtime: FormPropertyAttribute**

The **FormPropertyAttribute** is applied to a method in your control class, which allows an X++ method to be called as a **FormProperty** getter/setter from the control’s JavaScript class. A method with this attribute applied is called a **property**. Only use the **FormPropertyAttribute** on those X++ methods which need to be accessed directly from the control’s JavaScript class. The **FormPropertyAttribute** has no effect on the behavior of the X++ method when the method is used from within X++. Every property exposes an endpoint to the browser. As such, every property should be threat modeled and tested for exploits. The underlying X++ method should be declared private so that it is not accessible from other X++ code. If other X++ code needs to access the property, then declare a separate public X++ method without the **FormPropertyAttribute**, and move the shared property logic to this method. Then call this method from the private X++ method with the **FormPropertyAttribute**. This practice allows the property to perform logic that is specific to calls coming from JavaScript (such as argument type deserialization, argument validation, security validation, etc.) before executing the core shared X++ logic. The underlying X++ method must accept and return the desired type of the property. If the desired type if an EDT, the property must accept and return the base type of the EDT. The supported property types are:

- X++ primitive types
- X++ data contracts (whose members are also supported types)
- X++ List (whose items are also supported types)

You supply the following arguments to the **FormPropertyAttribute** constructor:

- **FormPropertyKind**: A required **FormPropertyKind** value that specifies the type of the property. Use **FormPropertyKind::Value** for Properties not bound to a data source field, and use **FormPropertyKind::BindableValue** for properties that may be bound to a data source field.
- **Name**: A required string that specifies the name of the property. A few best practices for naming properties:
  - Capitalize the first letter, and use PascalCase.
  - Do not use any of the names of inherited JavaScript properties
- **Read only**: An optional boolean that specifies whether this property is writable from the control’s JavaScript class. By default, properties have **Read only** set to **false**, so they are writeable. This argument does not affect the ability to write to this property from X++. To make the X++ method read only, remove all method arguments from the method declaration. A majority of properties should not be writable from the control’s JavaScript class. Because most property values require validation, a command should be used as a setter for
the property so that validation logic is can be run before the backing property is set.

- **Execute immediate**: An optional Boolean that specifies whether writes to this property are deferred or require immediate execution. By default, properties have *Execute immediate* set to \texttt{false}, so writes are deferred. Because the majority of properties should not be writeable form the control’s JavaScript class, the *Execute immediate* flag defaults to \texttt{false} and provides performance benefits. Even in the case of properties that are writable from the control’s JavaScript class, the performance side effects of immediate execution should be carefully considered before enabling the behavior.

The following example shows a typical property declaration. Most properties share the same boilerplate code for getting/setting, as shown below. The `textProperty` variable is the backing FormProperty field for this property.

```csharp
private str parmText(str _value = textProperty.parmValue())
{
    if(!prmIsDefault(_value))
    {
        textProperty.setValueOrBinding(_value);
    }
    return textProperty.parmValue();
}
```

**Runtime: FormProperty**

*FormProperty* is an X++ derived type used by the control framework for the synchronization of property values between X++ and JavaScript. *FormProperty* objects are considered the backing fields used internally by properties. Each *FormProperty* is typically used in 4 places throughout a control’s X++ runtime class:

1. The *FormProperty* is declared, usually right below the class declaration
2. The *FormProperty* is instantiated in the \texttt{new} method of the class
3. The *FormProperty* is initialized in the \texttt{applyBuild} method of the class
4. The *FormProperty* is read and written in the X++ method for the property

The following example shows a *FormProperty* being used in a typical controls’ X++ runtime class.
class MyControl extends FormTemplateControl {
    FormProperty textProperty;

    public void new(FormBuildControl _build, FormRun _formRun)
    {
        super(_build, _formRun);
        this.setTemplateId("MyControl");
        this.setResourceBundleName("/resources/html/MyControl");
        textProperty = this.addProperty(
            methodStr(MyControl, parmText), Types::String);
    }

    public void applyBuild()
    {
        BuildMyControl build;
        super();
        build = this.build();
        if(build)
        {
            this.parmText(build.Text());
        }
    }

    [FormPropertyAttribute(FormPropertyKind::Value, "Text", true)]
    private str parmText(str _value = textProperty.parmValue())
    {
        if(!prmIsDefault(_value))
        {
            textProperty.setValueOrBinding(_value);
        }
        return textProperty.parmValue();
    }
}

Runtime: new method

The `new` method on a control's X++ runtime class is called as a part of instantiating the control on a form. For the details on when the `new` method is called in the form lifecycle, please see the Control Lifecycle Diagrams. This method is used for instantiation of a control's FormProperties and setting the control's Template ID and Resource Bundle Path. See typical use of the `new` method in the example for FormProperty.

Runtime: applyBuild method

The `applyBuild` method on a control's X++ runtime class is called as a part of instantiating the control on a form. For the details on when the `applyBuild` method is called in the form lifecycle, please see the Control Lifecycle Diagrams. This method is used for initialization of a control's FormProperties to their default values, or to the values specified by the form developer who placed the control on the form. See typical use of the **applyBuild** method in the example for FormProperty.

Runtime: FormBindingUtil::initbinding method

The `FormBindingUtil` is an API provided by the control framework. It is used to bind FormProperties to data fields and data methods on a data source. The following example binds the data field with name "Value" on the data source with name "DataSource1" to the textProperty FormProperty of the runtime class.
Design time: The X++ build class

The build class defines the design time behavior of your control. This class determines which properties appear in the property sheet, and how the control behaves when it is modeled in the Form designer. The job of the design time class is to capture design time information for the runtime class to access later on.

Design time: Class declaration & FormDesignControlAttribute

The FormDesignControlAttribute is necessary for the control to appear in the Visual Studio Form designer when right-clicking on the design node of the form. If the FromDesignControlAttribute is missing, then the control can only be added to a form via imperative X++ code (i.e. via the addControlEx method on the form).

Design time: FormDesignPropertyAttribute

Placing this attribute on a method in the design time class will result in a new property with the name specified
by the first argument (and in the section specified by the second argument) appearing the property sheet for this control, with the corresponding X++ method operating as the getter/setter for the property.

```csharp
[FormDesignControlAttribute("MyControl")]
class MyControlBuild extends FormBuildControl
{
    str text;

    [FormDesignPropertyAttribute("Text", "Data")]
    public str Text(str _value = text)
    {
        if(!prmIsDefault(_value))
        {
            text = _value;
        }
        return text;
    }
}
```

**Design time: FormDesignProperty** **Attribute**

There are a number of FormDesignProperty attributes which may be applied alongside the standard FormDesignPropertyAttribute for specialized behavior in the property sheet. The specialized behavior includes enabling the property as a combobox which allows selecting from a list of values. The different types of lists that used are enumerated below. Whenever the user selects an item from the combobox, the string name of that item is passed into the X++ method getter/setter with the attribute.

- [FormDesignPropertyDataSourceAttribute] - Shows a list of the data sources on the form.
- [FormDesignPropertyDataFieldAttribute(<data source name>)] - Shows a list of the data fields on the specified data source.
- [FormDesignPropertyDataMethodAttribute(<data source name>)] - Shows a list of the data methods on the specified data source.
- [FormDesignPropertyFieldGroupAttribute(<data source name>)] - Shows a list of the field groups on the specified data source.
- [FormDesignPropertyExtendsClassAttribute(<class name>)] - Shows a list of the classes which extend the specified class.
- [FormDesignPropertyImplementsAttribute(<interface name>)] - Shows a list of the classes which implement the specified interface.
- [FormDesignPropertyReferenceAttribute(<FormDesignPropertyReferenceType::<type>])] - Shows a list of the specified AOT artifacts with the given type. The supported types are:
  - Table
  - View
  - Map
  - EDT
  - BaseEnum
  - Query
  - Class
  - Form
  - MenuItemDisplay
  - MenuItemOutput
  - MenuItemAction
  - Tile
  - KPI
The following example shows standard properties used to allow a Form developer to specify the Data Source and Data Field for the design time class.

```java
[FormDesignControlAttribute("MyControl")]
class MyControlBuild extends FormBuildControl
{
    str dataSource;
    str dataField;
    str dataMethod;

    [FormDesignPropertyAttribute("Data source", "Data"), FormDesignPropertyDataSourceAttribute]
    public str DataSource(str _value = dataSource)
    {
        if(!prmIsDefault(_value))
        {
            dataSource = _value;
        }
        return dataSource;
    }

    [FormDesignPropertyAttribute("Data Field", "Data"), FormDesignPropertyDataFieldAttribute(methodStr(MyControlBuild, DataSource))]
    public str DataField(str _value = dataField)
    {
        if(!prmIsDefault(dataField))
        {
            dataField = _value;
        }
        return dataField;
    }

    [FormDesignPropertyAttribute("Data Method", "Data"), FormDesignPropertyDataMethodAttribute(methodStr(MyControlBuild, DataSource))]
    public str DataMethod(str _value = dataMethod)
    {
        if(!prmIsDefault(dataMethod))
        {
            dataMethod = _value;
        }
        return dataMethod;
    }
}
```

A control with a design time class like the one above can then bind to the specified data source and data field inside of the applyBuild method, as shown below.

```java
public void applyBuild()
{
    BuildMyControl build;

    super();

    build = this.build();
    if(build)
    {
        this_parmText(FormBindingUtil::initBinding(
            build.DataSource(), build.DataField(), this.formRun(), build.DataMethod()));
    }
}
```

If you supply both a data field and data method to FormBindingUtil::initBinding, the data field binding will override the data method binding.
The following section documents the HTML attributes that are used in the control framework for control development.

**data-dyn-bind**

`data-dyn-bind`, the data binding attribute, standardizes many common DOM manipulations - such as modifying an element’s attributes, properties and CSS, or handling DOM events - through a declarative HTML-based API. The data binding attribute allows for these behaviors without requiring complex JavaScript. Using the data binding attribute rather than writing complex JavaScript can save the control developer valuable time by making things such as designing, debugging and maintaining the control much easier. However, complex JavaScript is still available when scenarios require its use. The data binding attribute binds HTML element behaviors to values supplied by the control developer. The values supplied can be simple JavaScript variables, JavaScript comparison or arithmetic expressions, JavaScript functions and JSON objects. The values supplied can also be observable variables, created using the APIs described in this document. The way in which the supplied value is bound to the HTML element is determined by the binding handler that is used with the data binding attribute. A list of all supported binding handlers is provided in this document. The data binding attribute requires the following syntax when used with any binding handler. The syntax for **data-dyn-bind** is:

```
data-dyn-bind="[first binding handler]: [value to bind to]"
```

The data binding attribute accepts a comma-separated list of binding handler-value pairs, so you can supply more than one binding handler to the binding attribute at a time. The following example binds the `visible` property of the div element to true, and binds the `textContent` property of the div element to "Hi".

```
data-dyn-bind="text: 'Hi', visible: true"
```

The data binding attribute is a custom HTML attribute understood by the control framework. The data binding attribute can be applied to any HTML element. Some HTML elements may not have the behavior which the binding handler modifies. For example, using the text binding handler on an `<svg>` element will not show the text since the `<svg>` element does not have a `textContent` property. The control framework reads and executes the data bindings specified in the control’s template at runtime. The lifecycle for the control in the browser can be summarized as follows:

1. The control’s HTM file is loaded by the browser.
2. Any script or resource files referenced in the HTM file are also loaded by the browser. Steps 1 and 2 are executed only once during a user’s session, even if there are multiple instances of the control.
3. The JavaScript class’s constructor for the control is call and passed with the X++ properties for the control instance.
4. The control’s template is copied from the HTM file and into the browser’s memory.
5. HTML elements in the control’s template are processed for data binding in hierarchical order (depth first), and data bindings on each element are executed in order from left-to-right.
6. The final HTML, including the original data binding attributes as well as any other markup added by the binding handlers or by the framework, is added to the browser’s DOM and rendered for the user to see.
7. Later, when the value of an observable changes, any binding handlers subscribed to the observable are re-executed and the live DOM is updated in real-time.
The `attr` binding handler applies the supplied HTML attribute and value to the element. For a list of HTML attributes see [W3 Schools – HTML Attributes](https://www.w3schools.com/tags/tag_attribute.asp). The arguments are passed in as an object array. Each argument is dual-valued. The first value is the name of the attribute, and the second value is the value of the attribute.

- **Name**: a string that specifies the desired name of the attribute to create.
- **Value or Expression**: a string that specifies the value to set on the attribute. If an expression is supplied, the value returned by evaluating the expression will be used.

The following example creates the title and name attributes and sets their value.

```html
<!-- the markup in the HTML template -->
<div data-dyn-bind="attr: {title: 'Hello', name: 'Greeting'}"></div>

<!-- the markup in the browser after the binding handler is applied -->
<div title="Hello" data-dyn-bind="attr: {title: 'Hello', name: 'Greeting'}" name="Greeting"></div>
```

The following example uses expressions and functions. However, using JavaScript functions as in-line HTML like the example below is not recommended.

```html
<!-- the markup in the HTML template -->
<div data-dyn-bind="attr: {title: false? 'Hello':'World', name: function(){return 'Greetings';}}"></div>

<!-- the markup in the browser after the binding handler is applied -->
<div title="World" data-dyn-bind="attr: {title: false? 'Hello':'World', name: function(){return 'Greetings';}}" name="Greetings"></div>
```

click

**Behavior**

Subscribes the supplied function to the click event on the element. For more information on subscribing to the click event see [jQuery – click()](https://api.jquery.com/click/).

**Arguments**

- `EventHandler (function)`

  The function to call when the event is raised.

**Example 1**

The following example shows an alert message "Hello" when the element is clicked.

```javascript
// In your control's code-behind JS file
<script>
... // boilerplate code
self.ElementClicked = function (event) {
  /* handle the click event */
  alert('Hello');
};
...
</script>

<!-- In your control's template HTM file -->
<div data-dyn-bind="click: $control.ElementClicked"></div>
```

The following example prevents the click event on child elements from bubbling up to parent elements. The example below will show only one alert with message “Hello” when the child element is clicked.

```html
<!-- the markup in the HTML template -->
<div data-dyn-bind="click: $control.ElementClicked"></div>

<!-- the markup in the browser after the binding handler is applied -->
<div data-dyn-bind="click: $control.ElementClicked"></div>
```
// In your control's code-behind JS file
<script>
... // boilerplate code
self.ParentElementClicked = function (event) {
    /* handle the click event */
    alert('Hi');
};

self.ElementClicked = function (event) {
    /* prevents the event form bubbling up to parent elements*/
    event.stopPropagation();

    /* handle the click event */
    alert('Hello');
};
...
</script>

<!-- In your control's template HTM file -->
<div data-dyn-bind="click: $control.ParentElementClicked">
    <div data-dyn-bind="click: $control.ElementClicked"></div>
</div>

The following example prevents the browser default behavior from executing. For anchor tags, the default hyperlink behavior is prevented, so the browser will not navigate to the link when the element is clicked.

// In your control’s code-behind JS file
<script>
... // boilerplate code
self.LinkClicked = function (event) {
    /* handle the click event */
    alert($dyn.format('Navigation to ' + {0} + ' was prevented', $(event.target).attr("href")));

    /* prevents the default event behavior */
    event.preventDefault();
};
</script>

<!-- In your control’s template HTM file -->
<a href="https://www.microsoft.com" data-dyn-bind="click: $control.LinkClicked">Click here</a>

CSS
Behavior
Adds or removes the specified CSS class name(s) to the element, based on the specified condition(s). Note that expressions supplied to the binding handler are only executed once.

Arguments
The arguments are passed in as an object array. Each argument is dual-valued. The first value is the Class name, and the second value is the Condition.

Class name (string)
The CSS class name to add to the element.

Condition (expression)
The condition on which to add the CSS class name. If the condition evaluates to true, the CSS class name is added. If the condition evaluates to false, the CSS class name is removed. If a supplied condition takes a dependency on an observable (via $dyn.value), then the condition will be re-evaluated whenever the observable value changes, and the associated CSS class name will be added/removed based on the new condition. The following example adds the CSS class names "red", "green", and "yellow".
// In your control's code-behind JS file
<script>
... // boilerplate code
self.red = function () { return true; };
self.yellow = $dyn.observable(true);
...
</script>

<!-- the markup in the HTML template -->
<div data-dyn-bind="css: {green: true, red: $control.red, yellow: $dyn.value($control.yellow)}"></div>

<!-- the markup in the browser after the binding handler is applied -->
<div class="green red yellow" data-dyn-bind="css: {green: true, red: $data.red, yellow: $dyn.value($control.yellow)}"></div>

event
Behavior
Subscribes the supplied event handler to the specified DOM event. For a list of supported DOM events, see jQuery - Event.

Arguments
For details on the arguments to the event binding handler, see jQuery - .bind(). The following example subscribes to the mouseover event and shows an alert when the element is hovered.

// In your control's code-behind JS file
<script>
... // boilerplate code
self.elementHovered = function (event) { alert($dyn.format('{0}',$(event.target).text()))};
...
</script>

<!-- the markup in the HTML template -->
<div data-dyn-bind="event: {mouseover: $data.elementHovered}">Greetings!</div>

foreach
Behavior
Repeats the content of the child element, updating the binding context of each child based on the supplied data. Supply only one child element inside of the element with the foreach binding. This one element is the element that will be cloned and repeated. Any other additional elements or content will be removed when the binding is applied. Binding handlers are executed in the order in which they appear on the element. Since the foreach binding changes the binding context, it is a best practice to always place the foreach binding after all other bindings on the element. This will ensure that preceding bindings are not affected by the binding context created by the foreach binding. To avoid performance issues, be careful to not create unintentional dependencies on observables inside of your foreach. Do not access an observable in the array using $dyn.value from within the child elements of the foreach, as the foreach binding has already subscribed to the observables in the array. Instead, use $dyn.peek to access an observable's value once without creating a subscription.

Arguments
Data (array list or JSON object)
The list of items to bind the child element to. If an array list is supplied, the binding context is an item in the array. If a JSON object array is supplied, the binding context is one of the object's properties.

Scope variables
When inside the scope of the foreach, the following scope variables are useful and can be used on the repeatable child element: $data, index, control, your own scope variables. The following example uses foreach to render a span element for each color in the array.
The following examples show a nested **foreach** binding. This example showcases how to use the index framework scope variable and custom scope variables to access the binding context from the parent element.
// In your control's code-behind JS file

<script>

... // boilerplate code

self.colors = [
    {
        Name: 'Red',
        Variants: ['Maroon','Burgundy','Sunrise']
    },
    {
        Name: 'Green',
        Variants: ['Sage','Forest','Lime']
    },
    {
        Name: 'Blue',
        Variants: ['Navy','Sky','Ice']
    }
];

</script>

<!-- the markup in the HTML template -->

<div data-dyn-bind="foreach: $control.colors">
    <div data-dyn-bind="vars: {$BaseIndex: $index, $BaseColor: $data.Name}"
        data-dyn-bind="foreach: $data.Variants">
        <div data-dyn-bind="text: $BaseIndex+'.'+$index+' '+$data+' '+$BaseColor"></div>
    </div>
</div>

<!-- the markup in the browser after the binding handler is applied -->

<div data-dyn-bind="foreach...">
    <div data-dyn-bind="vars...
        foreach...
            text...">1.1 (0.2552) X++ Language</div>
    <div data-dyn-bind="text...">1.2 (0.7) Applications</div>
</div>

<div data-dyn-bind="foreach...">
    <div data-dyn-bind="vars...
        foreach...
            text...">2. (600) Technology</div>
    <div data-dyn-bind="text...">2.1 (600.343) Microsoft Corporation</div>
    <div data-dyn-bind="text...">2.2 (600.117) Enterprise Resource Planning</div>
</div>

if

Behavior

Conditionally renders and binds the child elements of the element with this binding. This binding handler only operates on the child elements. It will not show/hide the element with the binding, nor will this binding show/hide the text content of the element with the binding. Bindings on the child elements will only be executed if the condition evaluates to true. Once the bindings on child elements have been evaluated once they will remain data bound even if the condition changes to false. This means that any calculations caused by bindings on child elements will continue to operate even after the child elements are hidden. Consider this when evaluating the performance of your control.

Arguments

Condition (expression)

Determines whether to render the children elements. The following example conditionally binds the show and text elements.
// In your control's code-behind JS file
<script>
... // boilerplate code
self.show = $dyn.observable(false);
self.text = "Hello";
...
</script>
<!-- the markup in the HTML template -->
<div data-dyn-bind="if: $control.show">
  <div data-dyn-bind="text: $control.text"></div>
</div>
<!-- the markup in the browser after the binding handler is applied -->
<div data-dyn-bind="if: $control.show"></div>
// Later on, the value of the “show” observable changes to true
<script>
... self.show(true);
...
</script>
<!-- the markup in the browser after the binding handler is re-applied due to the observable value changing -->
<div data-dyn-bind="if: $control.show">
  <div data-dyn-bind="text: $control.text">Hello</div>
</div>

sizing

Specifies the height and width of the control. The sizing binding handler should always be applied to the root element of the template (the element that has the id attribute), and supplied the height and width values from the X++ instance of the control by using the $dyn.layout.sizing helper function. See Example 1.

Arguments

The arguments are passed an object containing height and width properties.

Height (int)

Determines the height in pixels of the element on which the binding handler is applied.

Width (int)

Determines the width in pixels of the element on which the binding handler is applied. The following example specifies the size of MyControl.

<!-- this boilerplate binding ensures that the control’s container is sized based on the height and width properties -->
<div id="MyControl" data-dyn-bind="sizing: $dyn.layout.sizing($control)"></div>
<!-- the markup in browser after the binding handler is applied will vary based on the height and width properties defined in $control -->

The following example makes the control large or small depending on the value of the bigbox variable.

// Later on, the value of the “show” observable changes to true
<script>
... self.bigBox = true;
...
</script>
<!-- the markup in the HTML template -->
<!-- the markup in browser after the binding handler is applied -->
**Behavior**

Binds to the `textContent` property of the element. The text binding handler is meant to be used with UI text. It is not meant to bind non-string values (such as numbers, dates or Booleans) to the element. Convert all values into strings before supplying them to the binding handler, by using the `dyn.format` function. The text binding handler will replace all of the content inside of the element with the binding, whether or not the existing content is HTML or simple text.

**Arguments**

- `text (string)`

  The text to bind to. The following example binds the `textContent` property of the div element to the `text` property on the control.

  ```javascript
  // In your control's code-behind JS file
  <script>
  ... // boilerplate code
  self.text = "Hello";
  ...
  </script>
  <!-- the markup in the HTML template -->
  <div data-dyn-bind="text: $dyn.format('{0}',$control.text)"></div>
  <!-- the markup in browser after the binding handler is applied -->
  <div data-dyn-bind="text: $dyn.format('{0}',$control.text)">Hello</div>
  ```

**vars**

**Behavior**

Creates an HTML scope variable with the supplied name and value. The created scope variable is accessible only from bindings in the template. In addition, the scope variable is inherited by child elements. Binding handlers are executed in the order in which they appear on the element. Since the `vars` binding adds variables to the binding context, it is a best practice to always place the `vars` binding before all other bindings on the element. This will ensure that the subsequent bindings can access scope variables added by the `vars` binding. Do not create scope variables with any of the following names, as these names are reserved for framework scope variables: `$control`, `$data`, `$index`, and `$value`.

**Arguments**

- `Scope variables (object array)`

  The object array whose keys are the scope variable names and whose values are the initial values for the scope variables. The following example creates scope variables named "Hello" and "World" and displays their values.

  ```javascript
  <!-- the markup in the HTML template -->
  <div data-dyn-bind="vars: {$myVar: 'Hello', $myObs: $dyn.observable('World')}">
  <span data-dyn-bind="text: $dyn.format('{0} {1}!', $myVar, $myObs)">
  </span>
  </div>
  <!-- the markup in browser after the binding handler is applied -->
  <div data-dyn-bind="vars: {$myVar: 'Hello', $myObs: $dyn.observable('World')}">
  <span data-dyn-bind="text: $dyn.format('{0} {1}!', $myVar, $myObs)">
  Hello World!
  </span>
  </div>
  ```

**Example 2**

For an example, see the `foreach` binding handler examples.

**visible**

**Behavior**

Sets the visibility of the element. Always supply the visible binding handler on the root element of the template, and bind to the `Visible` property from the X++ control. This will ensure that the control respects the `Visible` property when it is set by a form developer or when it is set by the framework. If a control is initialized with its `Visible X++` property set to false, then the control will not appear on the form, and it the control’s template will not be loaded in the browser. If the control’s `Visible X++` property is set to true at a later time, then the control’s
Arguments

Visible (boolean)

Determines whether the element is visible or not. The following example sets the visibility of the control's outermost div element.

```html
<!-- the markup on the root element of the HTML template -->
<div id="MyControl" data-dyn-bind="visible: $control.Visible">Hello World!</div>
<!-- the markup in browser after the binding handler is applied -->
<div id="MyControl" style="display: none;" data-dyn-bind="visible: $control.Visible">Hello World!</div>
```

**HTML: Scope variables**

Scope variables can be used when binding values to binding handlers. Scope variables are only accessible from within the control's HTML template, and can only be used with the data binding attribute. Scope variables are neither accessible from other HTML attributes nor from the control's JavaScript class, but scope variables can be used in inline JavaScript expressions, functions and JSON objects that are passed to binding handlers.

*$control*

The $control scope variable provides the bindings in the HTML template with access to the properties and functions on the control's JavaScript instance. The following example binds visibility of the div element to the $Visible property of the control.

```html
<div id="MyControl" data-dyn-bind="visible: $control.Visible"></div>
```

*$data*

The $data scope variable provides elements with access to their current binding context. Only variables defined in $data (the binding context) or scope variables, can be used inside of HTML bindings. Variables that do not exist in the current binding context and do not exist as current scope variable cannot be accessed from an HTML binding. In most cases the binding context will be the control's JavaScript instance, so $data and $control will be equivalent. However, in some cases the binding context can change. For example, for elements inside of a foreach binding, $data provides the elements with access to the current array item. In cases involving multiple nested foreach bindings, elements in a nested binding may need access to the array item in a parent foreach binding. To access items in the parent foreach binding, you may create a scope variable which will be accessible to elements in the nested foreach binding. For an example, see the foreach binding handler examples.

*$index*

The $index scope variable provides a 0-based index of the array item when in a foreach binding. For an example, see the foreach binding handler examples.

**JavaScript: Inherited properties**

*Visible*

The Visible property is inherited from the base JavaScript class (via $dyn.ui.Controls.apply). There is also a Visible property in X++ that the runtime class inherits from the base FormControl X++ class. Simply bind this property to the visible binding handler and place it on the root element of the HTML template for your template will be loaded and instantiated in the browser at that time. A control's Visible X++ property can be inherited from its parent controls on the form. An element's visibility may be controlled by its parent elements, controls and containers, regardless of whether the visible binding handler is applied. The cascading nature of visibility is a standard HTML behavior and is not specific to the control framework.
Observable framework

$dyn.observe

Usage

Subscribes a function to changes of an observable. We recommend that you use dispose.

$dyn.observe(observable, observer, [context], [disposableObserver])

Arguments

Observable (observable)
Instance of an observable. Or a function, which will become a $dyn.computed.

Observer (function)
Function is invoked upon registration and also later when the observable is updated. Function is invoked with one argument, the value of the observable. If Observer returns false, then we un-subscribed automatically.

Context (options, optional)
Context to pass to the Observer. The Context becomes the this variable inside of the observer.

DisposableObserver (options, optional)
Unsubscribes the supplied DisposableObserver

Returns

Subscription (object)
Observable, ID, Dispose function (public) used to unsubscribe. The following example subscribes to the myObs observable, and executes the supplied function whenever the myObs observable value changes.

$dyn.observe(myObs, function (value) { console.log(value);});

The following example shows how a function can automatically subscribe to observables simply by accessing the observable using $dyn.value. The first function is treated like an observable whose value is dependent upon the value of two other observables (FirstName and LastName). Every time one of the observables (FirstName or LastName) changes its value, then the first function has also changed its value. When this happens, the second function (the callback function) will log the concatenation of the observable values to the console.

self.FirstName = $dyn.observable("Joanne");
self.LastName = $dyn.observable("Gordon");
$dyn.observe(
  function () {
    // Joann + " " + Gordon
    return $dyn.value(self.FirstName) + " " + $dyn.value(self.LastName);
  },
  function (value) {
    // "Joanne Gordon"
    console.log(value);
  }
);

The following example performs similarly to the previous example. However, this example uses a computed observable, named myComp, to handle the concatenation.
self.FirstName = $dyn.observable("Joanne");
self.LastName = $dyn.observable("Gordon");
self.myComp = $dyn.computed(function () {
  // Joanne + " " + Gordon
  return $dyn.value(self.FirstName) + " " + $dyn.value(self.LastName);
});
$dyn.observe(
  self.myComp,
  function (value) {
    // "Joanne Gordon"
    console.log(value);  
  },
  {FirstNameLabel: label1, LastNameLabel: label2}
);

$dyn.observable
Usage
Creates an observable variable.

$dyn.observable([initial value])
Arguments
initial value (optional)
The value to initialize the observable to.
Returns
Observable (function)
The newly created observable The following example creates and observable variable named "Hello".

var greeting = $dyn.observable("Hello");

$dyn.value
Usage
Accesses the value of an observable variable. When $dyn.value is called from inside of an observer function (such as an observer passed to $dyn.observe or $dyn.computed, as well as the binding expression passed to a binding handler) a dependency on the observable is created. This will cause the binding handler or callback to re-execute whenever the value of the observable changes. Because this dependency is created automatically when using $dyn.value, it is important to only use $dyn.value when you intentionally wish to create such a dependency. If you wish to access the value of an observable without creating a dependency, you should use $dyn.peek.

$dyn.value(observable)
Arguments
Observable
The observable property whose value to access.
Returns
Value
The current value in the observable property The following example returns the value of variable named observable and prints it to the console.

console.log($dyn.value(observable));

$dyn.peek
Usage
Accesses the value of an observable variable, without creating a dependency. For more information about dependency, see the $dyn.value function.
$dyn.peek(observable)

**Arguments**

Observable

The observable whose value to access.

**Returns**

Value

The current value in the observable The following example returns the value of variable named observable and prints it to the console.

```javascript
console.log($dyn.peek(observable));
```

$dyncomputed

**Usage**

Wraps a function with an observability scope. If observables are accessed from inside of the function by using the $dyn.value function, then the function will re-execute whenever the values of those observables change. Observables that are accessed by using $dyn.peek will not cause the function to re-execute when their values change.

$dyn.computed(observer, [context], [disposableObserver])

**Arguments**

Observer (function)

Observer (function)

Context (options, optional)

Context (options, optional)

DisposableObserver (options, optional)

DisposableObserver (options, optional)

**Returns**

Anything (optional)

If the Observer returns a value, then that value will also be returned by the call to $dyn.computed on the first time $dyn.computed is called (upon registration) as well every time the observer is invoked.

**Framework functions**

$dyn.callFunction

**Usage**

Calls the apply method on specified function. It cannot be used during an interaction.

**Arguments**

Function (function or observable)

The function to call. If an observable is supplied, the current value of the observable will be retrieved and used as the function.

This (object, optional)

The object to assign to this within the scope of the function.

Arguments (array, optional)

The arguments to pass to the supplied function.

Callback (function, optional)

The callback function to call when the supplied function has returned. The callback will be passed any values that are returned by the function that is called. The following example calls the apply function on the printName function.
var printName = function () {
    console.log(this.Name);
};
$dyn.callFunction(printName, self);

The following example calls the `getWholeName` function.

var getWholeName = function (first, middle, last) {
    var wholeName = first + " " + middle + " " + last;
    return wholeName;
};
var printName = function (wholeName) {
    console.log("Your name is: " + wholeName);
};
var firstName = "Joanne";
var middleName = "M";
var lastName = "Gordon";
$dyn.callFunction(getWholeName, null, [firstName , middleName, lastName], printName);

$dyn.format
Usage
Builds a string using the supplied values according to the supplied format.

Arguments
Format (string)
The format in which to build the string. Use bracket notation for placeholders.

Values (optional)
The comma separated values to use in the format

Returns
FormattedString (string)
The string after formatting has been applied

Example 1

var first = "Joanne";
var middle = "M";
var last = "Gordon";
var $dyn.format("Your name is : {0} {1} {2}", first, middle, last);

$dyn.label
Usage
Provides access to any labels stored via the Globalization API.

Arguments
Identifier (string)
The label ID, as specified to the Globalization API.

Returns
Value (string)
The label string in the current culture, if the Identifier is found. Otherwise, returns the supplied Identifier as a string. The following example returns and prints the label named “greeting”.

Globalize.addCultureInfo("en", {
    messages: {
        "greeting": "Hello!"
    },
});
console.log($dyn.label("greeting"));
CSS

Add namespaces to all CSS class names by prepending the class name with the control's template ID. This will prevent your control and its styles from conflicting with other controls in the client.

Flexbox

For advanced layout scenarios we encourage using Flexbox. Flexbox is compatible with the Extensible Control framework. Using CSS flexible boxes (Mozilla Developer Network) Please see the public Flexbox documentation for explanations and examples of the following topics:

- Responsive layouts
- Building columns and rows
- Arranging elements horizontally or vertically
- Arranging nesting elements
- Auto-sizing elements to stretch and shrink
- Locking/Freezing elements
- Building scrollable elements

Control Lifecycle Diagrams

Control Instantiation
This article describes the architecture that lets developers extend the user interface and also define new user interface patterns.

You can extend the existing application user interface (UI) and can also define entirely new UI patterns to create compelling new user experiences. By using modern tools such as HTML5, CSS3, and jQuery, developers can define customized visualizations of business data and drastically enhance the program’s interaction patterns.

Server-side architecture

The Control Extensibility Framework takes advantage of the existing and familiar X++ language for developing server-side data access and business logic. There are no artificial restrictions on the code that developers write to build extensible controls. Instead, developers can declaratively define the modeling experience and the run-time behavior through a set of X++ class and method attributes. A developer defines one class for the design-time behavior (the X++ Build Class) and one class for the run-time behavior (the X++ Runtime class).

- In the X++ Build Class, attributes enable the definition of design-time behaviors such as custom properties in the property sheet, the addition of child controls, and extra modeling components.
- In the X++ Runtime Class, attributes are used to define the run-time properties and commands that the extensible control will access from the client. The X++ Runtime Class consumes the X++ Build Class to initialize the run-time properties, based on the values and data bindings that are specified in the property sheet.

Client-side architecture

The client-side behavior for the control is defined by using HTML and JavaScript. In the context of a Model-View-ViewModel architecture, the HTML for the control is the View, and the JavaScript is the viewmodel. The Control Extensibility Framework provides an HTML-based binding syntax that enables elements in the HTML View to be bound to data fields and properties in the JavaScript viewmodel. In addition, the framework enables visualization behavior to be defined based on conditional expressions or logical evaluations that can react to changes in viewmodel properties or business data. The JavaScript viewmodel is automatically generated at run time, based on the properties and commands that are defined in the X++ runtime for the control. This automatically generated viewmodel lets a developer define an HTML View that consumes the properties and commands that are defined in X++. If a developer wants additional client-side properties and commands, or wants to implement visualization behavior that can’t be declaratively defined in the HTML View, the developer can extend the automatically generated viewmodel. The developer can take advantage of the JavaScript framework in conjunction with the powerful jQuery library.

Control extensibility architecture overview

The following diagram shows the artifacts that are involved and their relation to each other.
This article explains how to create localizable labels for client components and HTML/JavaScript controls.

This article details the process for creating localizable labels for client components and HTML/JavaScript controls. This process uses the existing localization tools and process for labels to bring localization support to client components and HTML/JavaScript controls. The following process relies on the label resource controller that can serialize label files into their JavaScript equivalents so that the labels can be used by the client components and HTML/JavaScript controls. The label resource controller is deployed automatically. It is an MVC service that is located at the /Resources/Labels endpoint.

1. Create a label file

Use the developer tools to create a new label file for your control's area, or use an existing label file for your control's area. A control's area is determined by the owning team.

- For extensible controls, your goal should be to create one label file for each HTM resource file. If multiple HTM resources share the same set of labels, only one label file should be required for the set of HTM resource files.
- For client controls and components, in general, controls that share a lot of the same functionality (for example, the Input controls: StringEdit, ComboBox, CheckBox, and so on) should also share the same label file.

Don't use a label file that also contains labels that are only used in X++. The whole label file is serialized when it's loaded by the client, so be sure to keep the labels that aren't required by the client components/controls in a separate label file.

2. Add label strings to the label file

Use the developer tools to add label strings to the label file. Example for extensible controls:

- **Label file name**: ClockControl
- **Label ID**: Seconds
- **Label string**: seconds

3. Request the label file as a JavaScript file by using Resource manager

Use the script loading tag to load the JavaScript file. The loading tag should reference the label file from /Resources/Labels, because the label resource controller is located there. Note: For extensible controls, the controller automatically appends the label file name to the beginning of the label identifier in JavaScript.

```html
<script src="/Resources/Labels/ClockControl.js"></script>
```

The JavaScript file that is returned will contain code that resembles the following example.
Globalize.addCultureInfo("en-us", {
  messages: {
    ClockControl_Seconds: "seconds"
  }
});

The culture information is injected automatically, based on the current user’s culture. No action is required on the part of the control to set, modify, or read the user’s culture. The preceding code adds each of your label strings to the jQuery Globalize label storage. You can then reference your labels throughout your HTML and JavaScript. The JavaScript code in the script file is run the moment that the file is loaded by the browser. Therefore, your labels are immediately ready for use. Be sure to add the label script loading tag after any other script loading tags in your HTML. The script loading tags are processed in order, from top to bottom. By loading the label script last, you help guarantee that no other scripts cause conflicts or override the labels that are set in the script label file.

4. Use localizable labels in HTML and JavaScript

The following framework application programming interface (API) can be used in HTML (inside data-dyn-bind) or in JavaScript to access the labels. HTML

```html
<!-- Example of using a localizable label with a Label Control. Supply the label to the "Text" property on the control -->
<div data-dyn-role="Label" data-dyn-bind="Text: $dyn.label('ClockControl_Seconds')"></div>

<!-- Example of using a localizable label with a basic HTML element. Supply the label to the "text" binding handler for the element -->
<div data-dyn-bind="text: $dyn.label('ClockControl_Seconds')"></div>
```

JavaScript

```javascript
/* Example of using a localizable label in JavaScript. */
var string mylabel = $dyn.label('ClockControl_Seconds');
```

You can also pass the label ID via a variable in HTML or JavaScript. HTML

```html
<div data-dyn-bind="text: $dyn.label($data.SecondsLabel)"></div>
```

JavaScript

```javascript
var string mylabel = $dyn.label(self.SecondsLabel);
```

The $dyn.label API will find the appropriately named label and return the string that can be used to display text to the user. This API will automatically select the label, based on the current user’s culture.

Troubleshooting

If you have correctly created a label file, and the label has been deployed, you should be able to load the JavaScript version of the label file directly from the browser. You can access the JavaScript label file by navigating to the home page in the client and appending the following path to the URL: /Resources/Labels/MyLabelFile.js, where MyLabelFile is the name of the label file without the language suffix. For a deployed label file that is named MyLabelFile.en-us, follow these steps:

1. Navigate to the home page, and sign in. (On one-box deployments, the URL of the home page is)}

https://usncax1aos.cloud.onebox.dynamics.com/en/
2. Make sure that the desired language has been set by going to **Options > Language and Region**. (You don't have to change the language if it's already set to the language that you want.) Now that the user's language has been set in the current session, the label resource controller will know what language to use when the label file is loaded.

3. To load the JavaScript version of the label file, navigate to the label file by adding `Resources/Labels/MyLabelFile.js` to the URL. (On one-box deployments, the whole URL is `https://usncax1aos.cloud.onebox.dynamics.com/en/Resources/Labels/MyLabelFile.js`.)

4. The corresponding label file will be JSON-serialized, and the browser will either show the text on the current tab or prompt you to download the `.js` file. If you download the file, you can then open it locally to inspect it.

If the browser doesn't find the file, you might have mistyped the name of the label, or you might not have deployed the label correctly. There is never a physical `.js` file for the label in the web folder `/Resources/Labels`. The `.js` file is dynamically generated by the label resource controller.

**Microsoft Visual Studio form previewer**

The form previewer isn't currently configured to load labels via the label resource controller. The form previewer will load only labels that are defined directly in the `.js` file for the code behind (located at `/Resources/Scripts`). Until the form previewer is updated so that it can load `.js` files from the label resource controller, make sure that you also define the labels in the `.js` file for the code behind of the control. The dependency on these labels will be removed in a future update.
This article provides guidelines that you should follow when you specify the layout and sizing of extensible controls.

Dos and don'ts for achieving the desired layout

- Don't use the layout classes on your control directly. (For example, layout-container and layout-horizontal are classes that you might see on controls in the DOM.) Instead, use the layout binding handlers to apply these classes. Internet Explorer uses a different layout framework, and to add some inline styles to elements, this framework requires the extra binding information that the handlers provide. Therefore, make sure that the classes are not hard-coded into the controls.
- Don't use absolute positioning (position: absolute and top/bottom/left/right positions) for elements that are children of a container that uses the layout binding handler. Absolute positioning of these elements prevents the CSS classes that are applied from laying things out correctly.
- Be careful about using width: 100% and height: 100%. These settings might not always work when they are combined with our layout CSS classes. If you want filling behavior, it's a good idea to use the $dyn.layout.Size.available option of the sizing binding handler instead.

Layout binding handlers

**Layout**

Used to apply ArrangeMethod and Columns attributes to containers Options: arrangeMethod, Columns, Children

**ArrangeMethod**

- $dyn.layout.ArrangeMethod.vertical
- $dyn.layout.ArrangeMethod.horizontalLeft
- $dyn.layout.ArrangeMethod.horizontalWrap
- $dyn.layout.ArrangeMethod.horizontalRight
- $dyn.layout.ArrangeMethod.none (No layout CSS classes should be applied to the element.)

**Columns**

- $dyn.layout.Columns.fill – Use this constant. Depending on the control, either “fill” and “balanced” columns are used.
- $dyn.layout.Columns.fixed – Use a fixed value, such as 1, 2, or 3.

**Children**

- Use $data.Children if the content handlers (append, replace) will be used to append children through this container. It should be used only if this control is a container.

Example usage:

```
```

**Sizing**

**Height/width**

- $dyn.layout.Size.available (SizeToAvailable) – Fill the available space in that direction.
• `$dyn.layout.Size.content (SizeToContent)` – Size to the contents in that direction.
• `$dyn.layout.Size.fixed (Manual)` – A fixed pixel value.

Example usage:

```html
  <!-- Your content goes here -->
</div>
```

Things to note about the sizing binding handler:

• When you use the sizing binding handler in combination with the `SizeToAvailable` option (`$dyn.layout.Size.available`), the layout binding handler must be used on the parent `<div>`/DOM element. This is how we know which axis to fill along (horizontal or vertical).

Should you use layout/sizing handlers or set the properties directly?

• If your extensible control inherits from a client control (such as Group or PivotItem), you won’t use the binding handlers directly on the `<div>` that is associated with that control, because the binding is already there. However, you might have to set the properties as you want them directly in the `data-dyn-bind` attribute.

Example:

```html
```

**FAQ**

**Is my control being laid out as expected?**

A good way to tell is to inspect the element and look for the desired classes.

• Layout classes to look for, depending on the options applied:
  - layout-container
  - layout-vertical
  - layout-horizontal

• Sizing classes to look for:
  - For `$dyn.layout.Size.available`, you should see either `fill-width` or `fill-height`.
  - For manual size, you should see either `fixed-height` or `fixed-width`.
  - For `$dyn.layout.Size.content`, there should be no extra classes, and the manual height/width should be specified inline on the element.

If these classes don’t appear as you expected, examine the usage of your binding handlers, and make sure that you’ve read through the list of dos and don’ts on this page.
This article describes how Task recorder determines what instruction label to generate for controls. It then explains how you can make sure that these labels are meaningful for the user.

Every control must have useful and meaningful instruction labels, so that the task guide, Microsoft Word document, and Help content meet Content Publishing standards for readability. We must first define two terms:

- **Control label** – The value that comes from the label property on the control.
- **Instruction label** – The label that a control instructs Task recorder to use when it's describing how to use that control (for example, “Click OK” or “In the First name field, enter ‘John’”).

When a control logs an event to Task recorder, three methods can be used to determine the instruction label that is shown to the user:

- As part of logging the Task recorder event, the control might specify an exact instruction label ID to use. As a best practice, here is how label IDs should be named: [client control type name]_[property or command name] For manual specification of an instruction label ID, see the code example later in this article (OptionalInstructionLabelIDOverride).
- If the control doesn’t explicitly specify an instruction label ID, Task recorder looks in the SysTaskRecorderLabel file to try to find an existing instruction label ID that fits the following naming syntax: [client control type name]_[property or command name] If an instruction label ID of this type is found, Task recorder uses it.
- If a label can’t be determined by using the preceding methods, Task recorder falls back to a more general-purpose instruction label. The general purpose instruction labels are in the SysTaskRecorder label file. There is one general-purpose instruction label for commands and another for properties.
  - Here is the general purpose instruction label for commands:
    - **Label ID:** CommandUserAction
    - **Label string:** %2 the %1
  - Here is the general purpose instruction label for properties:
    - **Label ID:** PropertySetValue
    - **Label string:** In the %1 field, enter %2

When Task recorder has determined which instruction label to use (via one of the preceding three methods), it strForms the label by using the following arguments:

- **%1** – This argument is the control label. Task recorder gets the control label by inspecting the label on the intractable. However, a control can override this label and provide its own label. See the code example later in this article (OptionalControlLabelOverride).
- **%2** – This argument is either the value (for properties) or the command name (for commands). This value will be the value that the control sent to Task recorder as part of logging the event. However, the raw data value can be ugly or meaningless to an end user. Therefore, a control can also provide a more user-friendly version of the value that Task recorder can display instead of the raw data value. See the code example later in this article (OptionalValueLabelOverride).
- **%3–%5** – These are command arguments and are used rarely. However, grids use them to record the row number, for example.
Case study

Let's use the Checkbox control as a case study for improvement. Currently, an instruction label isn't specified for the Checkbox via either method 1 or method 2 (see the previous section of this article). Therefore, the general-purpose property instruction label is used instead. If someone records selecting the Checkbox for a field that is named **Show infolog on failure**, the Task recorder output currently looks like this:

| In the Show infolog on failure field, enter True. |

However, typical end users might not know what it means to set a Checkbox to **True**. Therefore, a suggested improvement is for the Checkbox to produce a label that looks like this:

| Check Show infolog on failure. |

To make this improvement, someone must add a new label ID. That user must then use the label ID when the event is logged to Task recorder by using method 1:

- **Label ID**: Checkbox_Value
- **Label string**: "%2 %1."

This change produces output that looks like this:

| True Show infolog on failure. |

This isn't quite what we want to see. Therefore, in addition, when the Checkbox logs the property change event to Task recorder, it should pass in a specific **value label** that says either “Check” or “Uncheck.” If the control explicitly specifies a value label, Task recorder will use that value label instead of the raw data value that was recorded (**True** or **False**, in this case). See the code example later in this article (**OptionalValueLabelOverride**). After the user creates the new label and specifies the value label when the event is logged to Task recorder, the control will have suitable text output:

| Check Show infolog on failure. |

Finally, the Checkbox should have a second instruction label for “example value” usage of the Checkbox. “Example value” represents a special way that Task recorder displays an instruction label to the user. The author of the task recording can specify whether the task guide should instruct users to enter the same values that were entered when the guide was recorded, or whether the task guide should instruct users to enter their own values that are specific to their business requirements. Example value instruction labels have the following label ID syntax: [client control type name]_[property name]_Example For the Checkbox example, the value label would look like this:

- **Label ID**: Checkbox_Value_Example
- **Label string**: "Check or uncheck the %1 field."

The following code example shows how property change events are logged to Task recorder by an X++ control. Similar application programming interfaces (APIs) exist for C++ kernel controls. Command events have a similar API.
[FormPropertyAttribute(FormPropertyKind::Value, #MyPropertyName)]
public str value(str_value = valueProperty.parmValue())
{
    if(!prmIsDefault(_value))
    {
        using (var scope = SysTaskRecorder::addPropertyUserAction(#MyPropertyName, this, _value, [OptionalInstructionLabelIDOverride], [OptionalValueLabelOverride], [OptionalControlLabelOverride]))
        {
            // Property set logic goes here
            valueProperty.setValueOrBinding(_value);
        }
    }
}
This topic provides detailed information about workspaces and the patterns and subpatterns that are used to build operational workspaces.

A workspace is defined as...

- Part of the primary navigation mechanism.
- A form that supports a business activity (a logical group of tasks that make up the work of a target persona).
- A way to provide an initial overview and to increase productivity in the activity by allowing simple tasks to be completed directly in the workspace.

Workspaces have the following goals...

- Enable the user to understand the current state of the activity to support informed decisions.
- Let users navigate to deeper pages by selecting data, which avoids round-trips to pages with no information.
- Let users perform light tasks in the workspaces to avoid round-trips to deeper pages.
- Complete an activity without leaving the workspace.
- Reduce the need for navigation.
- Provide visual impact.
- Be constructed using prescriptive patterns and best practices that lead to minimal COGS and fast response times.

To accomplish these goals, the operation workspace pattern was developed.

Examples

An example of a workspace is the Reservation management workspace in Fleet management. You can get to it by accessing the menu item FMClerkWorkspace. The workspace, shown above, has the following items:

- **Summary** - Contains tiles and a chart.
- **Rentals** - Contains a vertical tab control having three pages - the first is selected, and you can see the corresponding content on the rightmost side.
- **Statistics** - Contains stacked charts.
- **Related links** - Contains a series of grouped menu item links to forms of relevance to this user and activity.

The overall form, as well as each of the sections within, are defined using UX patterns and subpatterns. The corresponding patterns are described in detail in the following sections.
Patterns and subpatterns

When building an operational workspace, you must use the patterns and subpatterns defined for that purpose. These patterns and subpatterns are described below. In general, when a control is cited within a pattern's structure, it will be described as: Common name (ControlType) [cardinality] If cardinality isn't specified, then the item is required exactly once. Simple patterns and subpatterns have the structural tree omitted.

Patterns
There are the top-level patterns for use with operational workspaces.

Workspace Operational
This pattern is the primary operational workspace pattern and should be applied to the Design node of the operational workspace form. It will prescribe the following structure:

- Design
  - ActionPane (ActionPane) [0..*]
  - Workspace Page Filter Group (Group) [0..1]
    - Subpattern: Workspace Page Filter Group
  - Panorama (Tab)
    - Section Summary Tiles (TabPage)
      - Subpattern: Section tiles
    - Section Tabbed List (TabPage)
      - Subpattern: Section Tabbed List
    - Section Charts (TabPage) [0..1]
      - Subpattern: Section Stacked Chart
    - Section-Related Links (TabPage)
      - Subpattern: Section-Related Links

Form Part Section List
This pattern is used for Form Part forms containing a list. These lists are referenced within the Section Tabbed List TabPage in the Workspace Operational pattern.

- Design
  - Header Group (Group) [0..1]
    - Subpattern: Filters and toolbar - Inline OR Subpattern: Filters and Toolbar - Stacked
  - Content Grid (Grid)
  - Default Action Button ($Button - any type of button) [0..1]
  - See All Menu Item (MenuFunctionButton) [0..1]

Hub Part Chart
This pattern is used for Form Part forms containing a chart control. These chart forms are referenced within two subpatterns: Section Tiles and Section Stacked Chart. It requires exactly one Chart control.

Subpatterns
Workspace Page Filter Group
This subpattern is referenced in the pattern Workspace Operational, in Workspace Page Filter Group. It allows for a single input control, which can be used to filter the workspace as a whole.

Section Tiles
This subpattern is referenced in the pattern Workspace Operational, in Section Summary tiles. It allows for both tiles and charts. Any number of tiles and charts can be defined, in any order. Tiles are defined with TileButton controls, and charts are defined with Form Part controls. A chart Form Part must have dimensions that match those of a normal tile, to ensure the chart flows correctly with the tiles displayed.
**Section Tabbed List**
This subpattern is referenced in the pattern Workspace Operational, in Section Tabbed List. It allows for multiple list containers to be modeled, each of which ultimately references a Form Part that points to a form containing the desired list. It requires the following structure:

- Tabbed List (Tab)
  - Tabbed List Page (TabPage) [0..*]
    - Form Part Section List (FormPartControl). **Note**: The referenced form must use the form pattern, Form Part Section List.

**Section Stacked Chart**
This subpattern is referenced in the pattern Workspace Operational, in Section Stacked Chart. It allows for up to two charts, which will be rendered in a vertically stacked orientation. It requires the following structure:

- Tab page (TabPage)
  - First Chart FormPart (FormPartControl) [0..1]
  - Second Chart FormPart (FormPartControl) [0..1]

**Section-Related Links**
This subpattern is referenced in the pattern Workspace Operational, in Section-Related Links. It allows for a series of links, with one level of nesting permitted. It requires the following structure:

- Tab page (TabPage)
  - Menu Function Button (MenuFunctionButton) [0..*]
  - Links Group (Group) [0..*]
    - Menu Function Button (MenuFunctionButton) [1..*]

**Filters and Toolbar – Inline**
This subpattern is referenced in the pattern Form Part Section List, in Header Group. It allows for some filters and a toolbar, all on the same line. It requires the following structure:

- Group (Group)
  - Filter Group (Group) [0..1]
    - Quick Filter (QuickFilterControl) [0..1]
    - Custom Filter Fields ($Field – any type of field) [0..*]
  - Toolbar (ActionPane) [0..1]

**Filters and Toolbar - Stacked**
This subpattern is referenced in the pattern Form Part Section List, in the Header Group. It allows for some filter fields on one line, and a toolbar on a line below those filters. It requires the following structure:

- Group (Group)
  - Filter Group (Group) [0..1]
    - Quick Filter (QuickFilterControl) [0..1]
    - Filter Field 1 ($Field - any type of field) [0..1]
    - Filter Field 2 ($Field - any type of field) [0..1]
  - Toolbar (ActionPane) [0..1]

**Future best practices check**
There are a few best practice (BP) checks that will eventually be built for workspaces. These checks are intended to provide guidance to the user about items that are recommended for performance or design reasons.

**Filters are covered by indexes**
This check is intended to ensure that any field that is modeled for use as a field on a workspace-wide filter is covered by an index. This check will help ensure that performance remains superior when users are taking
advantage of these filters.

**Chart parts only contain OLAP charts**
When a workspace contains a chart, that chart is modeled as a separate form, and that form is referenced on the workspace via a Form Part control. The intent of this check is to ensure that any such charts ultimately referenced from a workspace are only using OLAP data.

**Count tiles have queries defined**
A tile caching system has been implemented to improve performance of workspaces, as these forms generally contain several count tiles. For these count tiles to work correctly with the caching system, each tile must have a query defined. That query may be defined on the tile or on the menu item referenced by the tile. The intent of this BP check is to ensure a query is defined in one of these two locations for all count tiles.
It's important that workspaces perform well, and that they be responsive (that is, the data that appears in a workspace is refreshed as expected and kept up to date). This topic discusses framework support for caching data that is used for tiles and lists.

Introduction

Workspaces are intended to be the hub of activity for most users. They display a wealth of information that is collected from various sources. Therefore, you must make sure that workspaces perform well and are responsive (that is, the data shown in the workspace is refreshed as expected and/or kept up to date). Caching can help guarantee great workspace performance when data comes from poorly performing queries, and is especially useful when multiple users require access to the same set of data at the same time. For example, if a grid on a form uses a complex (that is, low-performing) query, you can cache the results so that they are available for subsequent loads of the form. This article discusses framework support for caching data that is used for tiles and lists. In terms of a responsive workspace, when users navigate to a workspace (or return to it), they expect that the data in the workspace will be relatively up to date. For example, if a user takes an action that changes the data for a list or tile in a workspace, the workspace should reflect that change immediately after the action is performed (if the action was taken directly from the workspace) or when the user returns to the workspace form (if the action was taken on a different form). This article describes techniques for making sure that this type of data refresh occurs (both metadata and code) for both tiles and lists.

Count tiles

Automatic data caching

The framework automatically sets up data caching behind any defined count tile. Therefore, no extra code is required in this case. A Refresh Frequency metadata property on Tiles determines how often the count on the tile is automatically updated. The following table describes the options and guidance for this property.

<table>
<thead>
<tr>
<th>VALUE</th>
<th>WHEN TO USE</th>
</tr>
</thead>
<tbody>
<tr>
<td>As Fast As Permissible (5 seconds)</td>
<td>Query execution time is 25 ms or less, and there is demand to see the updated value all the time.</td>
</tr>
<tr>
<td>10 Minutes</td>
<td>Query execution time is less than 250 ms, and updated data must be seen periodically</td>
</tr>
<tr>
<td>24 Hours</td>
<td>Query execution time is less than 2000 ms, and/or the count isn't expected to change often or up-to-date counts are vital.</td>
</tr>
</tbody>
</table>

Refresh management

Count tiles that show values that represent pending work should be fairly responsive. Ideally, the fastest possible refresh frequency should be defined for these tiles, so that the count that appears on a workspace tile is always within five seconds of being up to date. Because this quick refresh rate should be set only on performant queries (queries that have an execution time of less than 25 ms), the first recommendation is that you work on query performance for count tiles, to try to get query execution under the 25-ms threshold. In general, achieving this execution speed requires two things:

- A selective WHERE condition on the query
- Preferably, the conditions on the query should reduce the result
set to fewer than ~500 records, and ideally to fewer than ~100 rows.

- An index structure that can act on the selective WHERE condition

See the "Common mistakes and tips for query optimization" section for details about how to evaluate and improve query performance to achieve these execution speeds. For tiles that have backing queries and can't meet the 25-ms execution speed threshold, the refresh frequency on the tile should be set to one of the lower values (for example, 10 minutes or 24 hours). If the values must be updated more frequently for tiles that have less-efficient queries (which should be rare), you can add the following code to manually refresh the cache when an action is taken that will affect the cached set.

```cpp
TileDataService::forceRefresh(tilestr(<tileName>), formRun)
```

An example of a data set that might not change often is products that have no configuration. A tile that shows this count might have a refresh frequency of 10 minutes. However, the tile count might still appear responsive if the products form is instrumented to force-refresh the data cache when a configuration is defined for a product that previously had no configuration.

**Workspace lists**

Although the framework automatically sets up data caches for count tiles, manual setup is required for lists that must use data caching. Here are the high-level steps for introducing cached data for a list:

1. Create a query that references all the columns that you want in the cached data set.
2. Create a table that contains all the fields that you want to cache.
3. Create a class that defines a mapping between the cache query and cache table.
4. Add/reference the cached table as a data source on your form.

Each of these steps is described in detail in the following sections and is paired with an example from the Fleet workspace (Reservation Management).

**Cache query**

First, you must create a query that will be used to populate the cache table. This query should have the following characteristics:

- It should be against the tables that you want to get your cache data from.
- It should limit the results to the results that you’re actually interested in.
- It should select only the fields that you want to cache. **Note:** Each data source should have `DynamicFields=No` to avoid extraneous fields.

**Cache table**

Next, you must define a table that contains a set of fields that match the fields from the cache query. In addition to those fields, you must also add a field that is named `SysDataCacheContextId` (Int64). This field is used to map the cache row to the base cache tables. You’ll define a mapping on the table, between the `SysDataSetCacheTableMap` table's `Id` and `SysDataCacheContextId` fields and the cache table's `RecId` and `SysDataCacheContextId` fields, respectively. You can also define relations between this table and others, in addition to data methods that use the cached fields.

**Cache class**

The third step is to create a class that defines the relationship between the cache query and the cache table. This class requires that a few attributes be defined, and it must also extend and implement the appropriate framework data caching classes. The following code shows the corresponding class from the Reservation Management workspace.
In some circumstances, you might also have to implement the `parmQueryableToCacheMapping()` method. This method is required when at least one column name in your cache table doesn’t match the name of the corresponding column in the backing table (for example, if you must add two fields that have the same name but are from different tables). In this case, you can implement this method to define the column mapping between the cache table and the backing tables. The syntax is the same as the syntax for the `Query::Insert_RecordSet()` method (https://msdn.microsoft.com/library/query.insert_recordset.aspx).

```
public Map parmQueryableToCacheMapping()
{
    Map sourceToTargetMap = super();
    return sourceToTargetMap;
}
```

**Form implementation**

After you’ve built your cache query, table, and class, you’re ready to use the cache on your form. Add the cache table to your form as a data source, and reference it just as you would reference any other data source. In order for the form to correctly take advantage of caching, some code is required.

1. In a `registerDatasourceOnQueryingEvent()` method, add an event handler to the data source’s `OnQueryExecuting` event that calls `prepareDataSet`. This requires that the form class implement `SysIDataSetConsumerForm`.
2. If you want the form to use filtering, as provided by a workspace-wide filter, in a `registerDatasourceOnQueryingEvent()` method, register `applyFilter` on the `OnQueryExecuting` event. This requires that the form class implement `SysIFilterConsumerForm`.
3. If the form must react when a parent form’s filters change (for example, a workspace-wide filter), implement `SysIFilterEventHandler` on the form class, and include an `onFilterChanged()` method that calls `executeQuery()` on the cache data source.

The following code is an example from the `FMPickingUpTodayPart` form, which is one of the tabbed lists on the Reservation Management workspace.
Additional examples

In addition to the previous examples, you can refer to `SysFoundationTestOpenHeaderDataset`, which is used on `SysFoundationTestOpenHeadersFormPart` (which is itself referenced in `SysFoundationTestWorkspace`). This form uses the same method that is described earlier, but also includes some caching of a query aggregate count.

Refresh management

Lists of data in workspaces should also be responsive to user actions, especially if those actions cause records to no longer meet the criteria for appearing in the list. For example, you have a list of car rentals that are scheduled to start today. Every time that an employee initiates a rental record from that list with a customer, that record should no longer appear in the workspace list. Two mechanisms are available for keeping this list up to date:

- If the user takes an action that removes the record from the list, code can be added to guarantee that the list is immediately updated by refreshing the data source. If the data source is a cache data source, and the refresh frequency is slow, you might want to delete the corresponding records from the cache data source before you call `refresh`, to avoid having to force-refresh the cache.
- If the first option doesn't meet your requirements, you can asynchronously refresh form parts at a time interval. This option can be set up by using the `AutoRefreshInterval` property on the FormPart control.
  - For workspaces that have user interaction, the cache refresh frequency should be set based on query performance. In this case, the current recommendation is to not set the `AutoRefreshInterval`. Instead, rely on the user to manually refresh the workspace to bring in more data. Alternatively, you consider using an infrequent auto-refresh. However, in this case, the user's current selection will not be retained if the list is updated while the user is interacting with it.
  - A small percentage of workspaces are intended for viewing purposes only (no user interaction). These workspaces should use set the `AutoRefreshInterval` property programmatically to match the current refresh frequency of the backing cache. (Use `SysIDataCacheConfiguration.parmRefreshFrequency()` to retrieve the current refresh frequency of a cache, because it can be modified by the system administrator at run time.)
  - FormParts that have Charts should set the `AutoRefreshInterval` property to periodically show updated chart data.

Common mistakes and tips for query optimization

Here are a few general guidelines to consider when you optimize queries. This isn't an extensive guide to query optimization but provides just a few simple guiding principles.

- **Trace the statement by using SQL Profiler.** Attach the Microsoft SQL Server Profiler to the database, and capture the SQL statement that you want to optimize. You can use the tuning template that is provided in a default installation. Don't forget to disable tracing after you've obtained the statement that you're interested
Always look at the query plan. In Microsoft SQL Server Management Studio, make sure that you've enabled Include actual Execution Plan. Look at the query plan, and watch out for any warnings. The thickness of the arrows indicate how many rows have been fetched and brought to the next step.

Compare CPU milliseconds and logical I/O. One good way to determine whether a change to a given SQL statement improved the statement is to look at the logical I/O and CPU milliseconds. To obtain these numbers, use the following statements in the query editor:

- set statistics time on
- set statistics io on

Always clear caches when you measure a statement. To make sure that Microsoft SQL Server isn't using a cached execution plan, it's advisable that you flush the cache when you rerun a statement. To flush the cache, run the following two commands:

- dbcc dropCleanBuffers
- dbcc freeProcCache

Here are some common patterns to watch out for:

- Missing index: Analyze your WHERE conditions, and make sure that a selective index exists.
- Processing the same table/row many times: Especially when you do joins, try not to repeat yourself. Minimize the number of times that the same table/row must be processed. If you have tables that must be part of the result set but aren't used to narrow down the result set, move them as far out as possible (especially in union queries).

Test on volume data. To identify issues in your query, always run it on volume data.
This topic describes how to use Task recorder to record business processes.

Overview

**Task recorder**

Task recorder for Finance and Operations apps is a utility that lets users record business processes for several different use cases. Here are some examples:

- Step-by-step guided tours of a specific business process in the application itself
- Documentation of a business process as a Microsoft Word document that can optionally include screenshots
- Regression tests for a business process
- Automatic playback of a business process in the application

Task recorder for Finance and Operations apps boasts high responsiveness, a flexible extensibility application programming interface (API), and seamless integration with consumers of business process recordings. Task recorder is also integrated with the Business process modeler (BPM) tool in Microsoft Dynamics Lifecycle Services (LCS), so that users can continue to organize their recordings. However, users can no longer produce business process diagrams from recordings.

Task recorder can automatically generate application regression tests from business process recordings and playback previously recorded processes. These features also include test-specific gestures that let users take full advantage of Task recorder.

**Architecture**

Task recorder can record user actions in the client with exact fidelity, because every control is instrumented to notify Task recorder about the execution of user actions. The control notifies Task recorder that an event has occurred and passes all the relevant information about the corresponding user action in real time. From this information, Task recorder can capture the type of user action (for example, a button click, value entry, or navigation) and any data that is related to the user action (for example, the input data value and type, form context, or record context). Task recorder persists the information with enough detail to ensure that a playback of the recording can run the recorded actions exactly as they were performed by the user.

**Basic configuration**

Task recorder is included with every Finance and Operations app, and lets users begin to record business processes immediately after they open the client for the first time.

---

**IMPORTANT**

The Task guides tab is currently not available in Commerce or Human Resources. We are currently working to enable this functionality in a future release. Task guides in the Getting Started experience in Human Resources remain available to cover basic functionality. Procedural help is also available at Finance and Operations application documentation for both Commerce and Human Resources.

---

**Start a new recording**

The following steps show how to use Task recorder to start a new recording.
1. Open the product, and sign in. It's a good practice to refresh the browser before each new recording. A refresh creates a new user session and restarts Task recorder. Therefore, it provides the most stable recording experience.

2. Select the company that you want to use while recording. If this is your first time using Task recorder, you can follow along as this tutorial creates a sample recording based on a Fleet Management business process. You will need to load the Fleet demo data to follow along:
   a. Go to Dashboard > Fleet Management > Fleet setup.
   b. Click Load demo data.
   c. When the data is finished loading, click Close.
   d. Go back to the Dashboard by clicking the product name in the navigation bar.

3. Go to Settings > Task recorder.

4. The Task recorder pane is opened. You can click the Close button (X) in the upper-right corner to close the Task recorder pane before you begin a new recording. You can reopen the pane by following the previous steps.

5. Click Create recording.

6. Enter a name for the recording and click Start. Recording begins the moment Start is clicked. For the Fleet example in this tutorial, we'll use the name "Create a new rental reservation."

   While you're recording, you can click the Close button (X) in the upper-right corner to hide the Task recorder pane without stopping the recording. You can reopen the pane by clicking the Task recorder button that appears at the top of the page. This button appears only while recording is in progress.

   **NOTE**
   If the Saved views feature is turned on, recordings should be created by using either published views or the standard view, to ensure that recordings work reliably for users.

7. Task recorder enters recording mode. The pane shows information and controls that are related to the process of recording.

Now you're ready to record a business process using Task recorder. If you're following this guide as a first-time user, you may complete the following Fleet Management scenario as an example. Otherwise, you can record your own application scenario.

**Record a Fleet Management scenario**

1. In the Task recorder pane, click Start sub-task.
2. Set Name to "Create a new rental customer". Leave the Comment field blank.
3. Click OK. The task is added to the step list.
4. Go to Dashboard > Fleet Management > Reservation Management.
5. Go to All customers under the Summary tab.
6. In the Action Pane, click New.
7. Enter a first and last name for the customer.
8. Click Save.
9. In the Task recorder pane, click End sub-task.
10. Return to the Reservation Management workspace by clicking the browser back button twice.
11. In the Task recorder pane, click Start sub-task. Name the task "Rent a vehicle to the new customer". Click OK.
12. Click (+) Rental under Summary.
13. Under **Information**, select a “1975 Litware McKinley” as the vehicle.
14. Under **Information**, set the customer to the one just created.
15. Expand the **Discounts** section.
16. Click **Add** under **Discounts** and add the Frequent Customer discount. Click **OK**.
17. In the Action Pane, click **Start Rental**.
18. Set the return date to some date in the future.
19. Click **OK**.
20. In the **Task recorder** pane, click **End sub-task**.
21. Click **Stop** at the very top of the page.

**Recording a business process**

After you've started your recording, you can perform your business process just as you would typically perform it by using the web client. As you interact with the product, new steps are added to the step list in the **Task recorder** pane. In this section, you will learn about other actions that you can perform while you're recording a business process, to take full advantage of Task recorder's capabilities.

**Stop**

**Stop** is used to end the recording session. Before you click **Stop**, you should make sure that the recording is completed, because this action isn't reversible. When you click **Stop**, you're taken to the download options screen.

**Start/End sub-task**

**Start/End sub-task** lets a user specify the beginning and end of a set of grouped steps in a recording. Click the **Start sub-task** button to add a “Sub-task” step to the end of the current list of recorded steps. The sub-task will include all steps that you perform from this point until you click the **End sub-task** button. When you click the **End sub-task** button, an “End sub-task” step is also added to the list of recorded steps.

**NOTE**

You must start a sub-task before you perform/record the steps that you want to include in the task. Then, after you've performed/recorded all the steps that you want to include in the task, you must end the sub-task.

Sub-tasks are purely an organization tool, and consumers of business process recordings can interpret the task groupings in useful ways. Because tasks can be nested inside other tasks, they provide the flexibility to organize very long and complex business processes.

**Delete/Restore step**

**Delete/Restore step** enables a user to remove steps from the recording, or undo the removal of a step from the recording. You must first select the step in the Steps list that you want to delete/restore, and then click the **Delete/Restore step** button.
Enriching steps in a recording

There are various options for enriching a step in a recording. For example, you can adjust the text that is associated with a step and add information about a specific step. This section describes the step enrichment capabilities that are available. To access these options, click the Edit step button on a specific step of a recording.

Step instruction

The Step instruction is the primary text that is displayed for this step in the task guide. There are usually 2-3 alternative options for step instructions, and they appear in the following order when editing the annotation.

- In the First name field, type 'John'.
- In the First name field, type a value.
- In the First name field, { your example text }.

Steps that are not related to fields, such as clicking buttons, opening forms, or selecting records from a lookup, do not set Example value label as an option when annotating.
● **User-supplied value label** This step instruction contains placeholder text, which the author can fill in with their own text. For steps which have an **Example value label** option, the placeholder allows substituting the text which normally specifies the data to enter. This is useful for scenarios where neither the **Preferred value label** nor the **Example value label** sufficiently express the data that should be used for this step.

  ○ **Example label.** In the First name field, enter *(your example text).*
  ○ **Example label after supplying the placeholder text.** In the First name field, enter the customer's name.

For steps which do not have an **Example value label** option, the placeholder allows substituting all of the label text. Steps associated with buttons, for example, do not have **Example value labels**, so you may replace the entire label text with your own text.

  ○ **Example label before replacement.** Click Post.
  ○ **Example label after replacement.** To post the order, click Post.

**Titles and notes**

Titles and notes provide places for user-specified text to be associated with a step in a task guide.

- **Title** – The title lets you specify the text that appears above the step instruction for this step in the task guide. The title a good place to put text that you want users to read before they complete the action that is indicated by the step instruction.

- **Note** – You can use a note to specify text that appears in the expandable section of the pop-up for this step in the task guide. A note is a good place to put optional reading material or other information that might be useful to users, but that they aren’t required to read to complete the action that is indicated by the step instruction.

**Change recorded values**

Starting in version 10.0.12, you can adjust the values that are recorded in basic input controls (for example, simple text, numeric, date, and picklist fields), without having to re-record those steps. Note that lookup controls and reference groups aren’t currently supported.

**Hide from task guide**

The **Hide this step** option lets the author prevent specific steps from appearing in the task guide. This option is useful for hiding steps that are required for the task recording to run in playback mode, but that should not be seen by users. Examples of these steps include copy steps, system-generated steps, and data clean-up steps. If you hide a sub-task, all the steps that are recorded inside that sub-task will also be hidden.

**Using control gestures**

The basic recording capability lets a user record an end-to-end business process by using Task recorder, but without adding overhead to the process. In some circumstances, more advanced recording features can be used to create even richer business process recordings. Each of the following gestures is found under the **Task recorder** option on the shortcut menu (also known as a right-click menu or context menu) for a control and causes a step to be added to the recording. If the gesture isn't supported for a control, it won't appear on the shortcut menu for that control.

**Copy**

The **Copy** gesture lets you copy the value for the current control to the Task recorder “clipboard.” That value can then be used later as part of a **Paste** or **Validate** gesture. Because values from multiple controls might have to be pasted, the Task recorder clipboard maintains a list of all control values that have been copied in the recording.

**Paste**

The **Paste** gesture lets you paste a value from a previous **Copy** gesture in the same recording. The Task recorder
paste function works like the standard paste function that users might be familiar with, but it has an additional benefit when it’s used during recordings. Because Task recorder will replay the recorded Copy and Paste commands during playback, if the copied control has a different value than it had during recording, Task recorder will paste the current value instead of the value that the copied control had during recording. This feature is useful in scenarios where the copied control has a value that can change between environments (for example, recID values or number sequences).

There is an additional benefit from using the Copy and Paste gestures when test code is generated. For any control where the value is set via the Paste command, Task recorder doesn’t have to create a parameterized input variable for that control’s value, because it's set based on another control’s value. This feature can be very useful in scenarios where an entity such as a customer is created, and an identifier for that entity is frequently entered during the recording. Instead of manually re-entering the customer name or ID throughout the scenario, and causing Task recorder to generate a parameterized input variable for each entry, the user can copy the customer name or ID one time, and then repeatedly paste it. In this case, Task recorder will generate a single parameterized input variable to represent the customer name or ID. This feature can make it much easier to change the input data for a generated test.

**Validate**

The Validate gesture lets you insert a step that validates the value of the targeted control. This gesture always uses equality to validate the control value. **Validations aren’t currently run during recording playback.** Instead, they are run only when the generated test code is run. Two kinds of validation are available:

- **Current value validation** will capture the targeted control’s value at the time of recording and use it to generate an assertion in the test code. In the list of validation options on the shortcut menu, **Current value** is always first.
- **Reference value validation** will use the value of a previously copied control when generating an assertion in the test code. This allows creating assertions that are resilient to changes in the data, since the value is not hardcoded into the test code. In the list of validation options on the shortcut menu, **Reference value validation** follows the format [AOT name of copied control: current copied value].

Additional options are available in version 10.0.13 and later. Here are some examples:

- **Enabled/Disabled** validates that the targeted control’s state is enabled (or disabled), and then uses that validation step to generate an assertion in the test code.
- **Read-only/Editable** validates that the targeted control’s state is read-only (or editable), and then uses that validation step to generate an assertion in the test code.

**Add info step**

The Add info step gesture lets you insert a step and supply your own text for it. This feature is useful primarily for creating task guides. An informational step (or info step for short) is a task guide step where the instruction text for the step is user-specified. Info steps are useful for describing actions that are a part of the scenario but must occur outside the client. For example, a scenario might require the user to search for item inventory or check an email for information.

You can specify where an info step should appear in the task guide. The info step can point to a control on the page, if the step is associated with the control. Alternatively, the info step can appear in the upper right of the page, if the step is external to the client, or if it’s an explanation that applies to the whole page.

**NOTE**

Because info steps are manually specified steps and are not automatically recorded by Task recorder when the user takes an action on a control, the info step does not have the capability to automatically progress when a user completes the step in the task guide. Because the info step is not associated with taking an action in the client, there is no action for a task guide to detect that the user has completed in order to automatically progress to the next step.
Options after a recording is completed

After you click Stop to end your recording session, several options are shown so that you can save the files that are related to the completed recording. Select Save to this PC, and save the task recording package to your desktop. You will use this file later.

Save to this PC

One option after you finish your recording is to download the task recording package (an .axtr file) to your computer. This file can be loaded later via the Task recorder pane, so that the recording can be played as a task guide or edited.

Save to Lifecycle Services (LCS)

When you save your recording to an LCS library, it's published on the specified business process in a BPM library. If the selected LCS library is set as a Help library, you will be able to find the task guide for the recording by searching the Help menu.

NOTE

To be able to save a recording to an LCS library, the user must be in the Azure Active Directory (Azure AD) tenant that the environment was deployed from.

Export as Word document

The Microsoft Word document for your recording contains the recorded steps as well as any screenshots that were captured.

Save as developer recording

The raw recording file (developer recording) is useful for developer scenarios, such as test code generation and scenarios where RSAT is used.

Playing back a recording

The playback functionality of Task recorder can automatically run the steps of an existing recording by using the pages and values that were originally recorded. Playback mode can be used to update an existing recording if changes were made to the underlying application, and those changes altered the business process steps that are required for the scenario. It's important to remember that, in this mode, Task recorder simultaneously re-
records the steps and plays them back. When the playback is completed, a new recording is produced that reflects both the steps that were run from the existing recording and any new steps that the user manually performed. Any steps that aren't run either by the user or automatically by Task recorder aren't included in this new recording.

To play back an existing recording, follow these steps.

1. Refresh the browser tab.

   NOTE
   It's a good practice to refresh the browser before each new recording.

2. Open the Task recorder pane.

3. Click Playback recording.

4. Click Open from this PC to load a recording from a previously downloaded Task recorder package (.axtr file).
   - If you're reading this guide for the first time and following along, choose the “Create a new rental reservation” file that you downloaded previously.

5. Click Start.

When you play back a recording, additional actions are available in the Task recorder pane.

Play next pending step

Play next pending step runs the next step in the recording. This action is useful because it gives you more control over the playback speed when you want to analyze the effects of a single step. This action has a side-effect that it's important to be aware of. When you click Play next pending step, any open lookups, drop-down dialog boxes, or Action Pane tabs might be dismissed, because this action removes focus from those elements. For these situations, we recommend that you use Play all pending steps instead.

Play all pending steps

Play all pending steps begins sequential execution of the remaining steps in the recording, and continues until either playback is paused or all steps have been run. During playback, the Play all pending steps button is replaced by a Pause button that can be used to pause playback. If playback can't successfully run a step for any reason (for example, because it can't find a button that has been renamed), Task recorder will skip that step, and playback will automatically be paused. In this way, the user has an opportunity to replace the obsolete step by completing the new steps in the client. Task recorder will record the new steps and ignore the step that was skipped. The user can then click Play all pending steps to continue playback for the remaining steps. After the recording is completed, the user can download the updated recording. This recording will contain all the steps of the original recording, but will exclude any skipped steps and include any new steps.

Play to selected step

Play to selected step behaves like Play all pending steps, but it lets you play only a subset of the steps instead of all the steps. In the list, select the step that you want playback to stop at, and then click Play to selected step. Task recorder will begin to run the steps in the list and will stop when it has run the step that you selected.

Editing a recording

Although you can edit a recording through the playback functionality, there is also a mode that lets you make simple edits to a recording without having to replay the whole recording. To access this feature, click Edit recording after you open the Task recorder pane. You can use this feature to make the following edits:
- Insert steps into a recording without re-recording the whole file.
- Move steps under a sub-task without re-recording the whole file.
- Adjust the name and description of the recording.

**Insert steps without re-recording the entire file**

You can add a step anywhere in a task recording without playing back or re-recording the whole file.

1. Select the step after which you want the new step to be inserted. Make sure the step is highlighted.

   In order for task recorder to insert a step, you must have the correct page open. The correct page is the page on which the new step occurs. Task recorder has a mechanism that determines what the active page is, and will disable the functionality if the correct page isn’t open.

   ![Insert step](image)

   When you are on the correct page, **Insert step** becomes available.

2. Click **Insert step**.

   When you click **Insert step**, Task recorder switches to recording mode. Any action that is performed in the user interface (UI) will now be recorded and inserted into the recording as steps.

3. Click **Stop**.

   Recording mode is stopped, and you can now continue to edit the recording. For example, you can repeat this process to insert steps in other places in the recording, or you can move sub-tasks as described in the next section.

4. When you’ve finished editing the task recording, click **Done editing**, and then select one of the options to save or publish the recording.

**Move steps under a sub-task without re-recording the entire file**

You can move steps under a sub-task without playing back or re-recording the entire file. You can also move the sub-task step or the end sub-task step if you want to group an existing block of steps.

1. Select the step or sub-task step that you want to move. Make sure that the step is highlighted.

2. Click **Move step after**. To access this command, you might have to select the ellipsis (…) button.
3. Select the step or sub-task step that you want to move the step or sub-task step after. Task recorder will move the step.

4. To move the end sub-task step, select it, click **Move step after**, and then select the step that you want the end sub-task step to be after.

   If you want the first step in the task guide to be within a sub-task, create a sub-task step as the second step, and then move the first step into it. You can add or move as many steps or sub-tasks as needed.

5. When you’ve finished editing the task recording, click **Done editing**, and then select one of the options to save or publish the recording.

**Adjust the recording name and description**

You can adjust values of the **Recording name** and **Recording description** fields. If you want to see more steps in the Task recorder editing pane, you can also collapse the section that shows the recording name and description.

---

**Playing a task guide**

A **task guide** is a user-focused experience that lets the user follow a guided step-by-step set of instructions to complete a business scenario by using a task recording. The user is instructed to complete each step through an animated pop-up prompt that will move across the page and point to the UI element that the user should interact with. The prompt will also tell the user how to interact with the element. For example, it might state, “Click here” or “In this field, enter data.” Each step that the user is instructed to complete is based on a step that was originally recorded in the task recording. Because the task recording file contains the data that describes the step that was originally recorded, the task guide can automatically determine when the user has completed the step as expected. It then automatically moves on to the next step.

---

**NOTE**

One way that the task guide determines that a user has completed a step is by detecting when the value in a field has changed. Although the task guide doesn't require that a specific value be set, it does require that the field value be changed in order to determine that the step was completed. The user must change the field value, and then press the Tab key or click in an area outside the UI element. Only at that point does the client detect that the field value has changed, and it can then proceed to run any required application validation or business logic. Therefore, before the task guide can determine that the step was completed by the user, it relies on the client to detect that the field value has changed.
What can a task guide allow a user to do?

When a user is completing a task guide, the client behaves in the same manner, with the same data, security, and validation rules as it does when the user is not completing a task guide. There is no difference of behavior in the client that would allow a user to take an action that they cannot otherwise take when they are not completing a task guide. When a user is completing a task guide:

- Any data the user enters is subject to the same data validation rules as when not playing the task guide.
- Any data the user enters may be saved, and the user may modify data according to the same restrictions and rules as when not playing the task guide.
- Any security mechanisms the user encounters behave the same as when the user is not playing the task guide.
- Any forms or controls the user accesses are subject to the same security and access mechanisms as when the user is not playing the task guide.

The “On-rails” feature of task guides

By default, when a user begins a task guide, they are placed “on-rails”. These "rails" prevent the user from clicking on elements other than the element the task guide is pointing to. When the user tries to click on something outside of the UI element that the task guide is pointing to, the task guide pop-up will animate to let the user know that they cannot progress until they complete the current step of the task guide.

While a user is prohibited from clicking on other elements, the user is not prevented from tabbing through the other controls on the form, and the user is not prevented from using keyboard shortcuts. This is by design, as the “on-rails” feature is designed for and targeted at first-time users, who are expected to primarily use the mouse as they become familiar with the application.

More advanced or experienced users can turn off the “on-rails” feature when they complete a task guide. At any point during the task guide, these users can turn off the rails by clicking the Unlock button that appears on the Task recorder toolbar at the top of the page. This button can also be used to restore the rails at any point during the task guide. In some situations, the task guide might automatically turn off the “on-rails” feature. When the rails are turned off, the user can click UI elements just as they do when the task guide isn't running. The "on-rails" feature might be automatically turned off in the following situations:

- The user is being directed to go to a page by using the navigation pane or navigation search.
  - Because the user can use either entry point, the task guide doesn't point to a specific entry point, and it doesn't prevent the user from using either entry point.
- The task guide enters an error state (see the next section for a list of error states).
- The task guide is showing an info step.

Error detection

An error state occurs when the task guide is not able to point to the UI element that is associated with the current step because the UI element is not visible on the screen. When the task guide detects that the current step requires the user to interact with a UI element that is not visible, then the task guide pop-up will move to the upper-right side of the screen. These causes of an error state can be simplified into two categories.

The control is not visible on the form

This error state usually occurs when the user has opened or closed the incorrect tab, FastTab, collapsible section, FactBox, or pop-out menu.

Because the UI element that is needed for the current step is somewhere on the current form, but it is not visible on the screen, the task guide pop-up will simply move to the upper-right side of the screen while displaying the same instruction that informs the user of the action they need to take.

Because the task guide can't find the UI element on the screen, the user must manually determine what is causing the UI element to be hidden and then make the element visible on the screen. The task guide pop-up will automatically detect that the UI element is visible and will reposition itself so that it's pointing at the now-
visible element.

The control is not on the form

This error state usually occurs when the user has gone to the wrong form, either by navigating to the wrong form or by leaving the correct form.

Because the UI element is not visible on the screen, the task guide pop-up will move to the upper-right side of the screen. In addition, when the task guide detects the user is on the wrong form, the task guide pop-up text will change to inform the user of the form they should navigate to.

In some cases, the task guide pop-up will not mention the form by name. This is because the user may need to navigate to a dynamic form. A dynamic form is a form that is not modeled, frequently known as a runtime-generated form. These sorts of forms do not have a proper name. Some examples of runtime-generated forms include simple and custom lookups. The way for a user to navigate to a lookup form is to re-open the lookup.

Next step and Previous step

The Next step and Previous step buttons appear in the task guide pop-up and let a user manually control the flow of the task guide. When these buttons are clicked, the task guide will go to the next or previous step. The task guide doesn't verify that the user has completed a step before it goes to the next or previous step.

The task guide never automatically completes any step for the user, even when the Next step and Previous step buttons are used. Use of these buttons can cause an error state if the previous or next step refers to a UI element that isn't on the current page. When the user is completing an info step, the only way to proceed is to use the Next step button. This action is required because an info step doesn't represent an action that was recorded on any UI element. Because no action was recorded in the task recording, the task guide doesn't have the necessary information to determine what action it should expect the user to complete.

The See more button

When the See more button is clicked, the task guide pop-up expands to show additional information that is related to the step. The additional information is often optional reading material that isn't required for the user to successfully complete the step. The following information might be included:

- An Example value
  - The Example value is the value that was originally used when the task recording was created.
  - Example values appear only for steps that use non-lookup fields. These fields include text fields, number fields, date fields, combo boxes, and check boxes.
- A Note
  - A Note may contain scenario-specific information that will help provide context to the user about the current step of the task guide.

Taking screenshots in Task recorder

By using a pre-release Chromium browser extension that works for both the new (Chromium-based) Microsoft Edge browser and Google Chrome, Task recorder can take screenshots of the browser as a user records a business process. After the user completes the recording, Task recorder can use these screenshots to generate Microsoft Word documents. To turn on this functionality, follow these steps to install the pre-release Chromium extension that enables Task recorder to take screenshots during recording.

1. Download the FMLabTaskRecorderScreenshot folder that contains the extension from GitHub, at https://github.com/Microsoft/FMLab.
2. On-premises deployments only: Adjust the manifest for the extension so that it matches the following code. Replace <hostname> with the base URL for your environment.
Generating tests from a recording

After a business process recording has been completed by using Task recorder, a developer can import the raw developer recording file (.xml file) into Visual Studio to create an X++ test. The import tool generates a human-readable X++ test from the recording, and translates any control gestures, validations, or tasks into the appropriate test code.

**Import a recorded test**

1. Open Visual Studio by using the Finance and Operations development tools.

2. Go to *Dynamics 365 > Addins > Import task recording*.

3. In the *Import task recording* menu, use the *Browse* button to locate a previously downloaded recording file.

4. Optionally, choose to have the generated test code be added to the startup project. This requires that a solution containing a project is set as the startup project. This will place the generated X++ test into the

---

3. **21Vianet deployments only:** Adjust the manifest for the extension so that it matches the following code. Replace `.com` with `.cn`

```json
"content_scripts": [
    {
        "matches": ["https://*.dynamics.com/*", "<hostname>"],
        "js": ["screenshot.js"]
    }
]
```

4. Open the latest Microsoft Edge browser or Google Chrome.

5. Select *Settings and more > Extensions* in Microsoft Edge (or *Customize and control Google Chrome > More tools > Extensions* in Google Chrome).

6. Select *Developer mode*.

7. Click *Load unpacked extension*.

8. Browse to the folder that contains the Task recorder extension by using the path *FMLab-master > FMLab > TaskRecorderScreenshot*, and then select *Select Folder*.

9. Make sure that *Enabled* is selected so that extension is turned on.

10. Restart the browser.

Task recorder will now take screenshots of the tab where the client is running. These screenshots are available for one week after the recording has been played. (If you're running a platform version that is earlier than Platform update 16, the screenshots are available for only 15 minutes.) If the screenshots have expired, you can regenerate them by playing the task recording again.

Note that **Task recorder does not** capture screenshots from other tabs or of the user’s desktop.
same model as the project.

5. If you’re creating a new project, select the model for the project. The generated X++ test will be put in this model. For the generated test to be successfully built, the model must have references to the \textit{TestEssentials} model.

6. Click \textit{Import}.

7. In the \textit{New Project} dialog box, provide a name for the project.

8. After the project is created, the user can open and inspect the generated code.

9. To run the test, build the project.

10. Go to \textit{Test > Windows > Test Explorer}.

\section*{Appendix}

\textbf{Controls that are known to have incomplete support for Task recorder}

- Table
- Filter pane, which is the filter that pops out from the left side
  - When adding filters to the filter pane, the steps are delayed. The steps do not get recorded until the user clicks “Apply” on the Filter pane.
- Enhanced previews
  - No planned support for recording gestures inside of enhanced previews. While recording, enhanced previews will be disabled.
- No extensible controls are supported out of the box, except Segmented Entry.
  - Extensible control owners need to individually build support for Task recorder.

\textbf{Controls that can be recorded, but have limited support for the Copy/Paste/Validate gestures}

- Date/Time
  - Doesn’t support copy/pasting “Never” as a value.
- Image
  - No ability to copy/paste/validate an image value.
- Filter pane
  - Copy/Paste works, but the UI will not show the pasted data. You can proceed as if it pasted correctly.
- Message box
  - You cannot validate the text in the message box.

\textbf{Controls that are known to have incomplete support for being used in a task guide}

- Quick Filter, which is the filter control that appears above lists
  - Does not support displaying a “generic value” during the task guide. Currently displays the value that was used during recording.
- Filter pane, which is the filter that pops out from the left side
The task guide does not point to the individual elements within the Filter pane that need to be clicked
on.


This article provides a quick reference sheet that explains what each button in the Task recorder menus does.

Main menu

- **Create recording**
  Choose this option to begin creating a new recording.

- **Play recording as guide**
  Choose this option to see what your recording looks like when viewed as a Help topic or played as a Task guide.

- **Edit recording**
  Choose this option if you need to change the recording’s name, description, or the text that is displayed in the steps.

- **Playback recording**
  Choose this option if you need to add or remove steps. You can also use this mode to automatically play a recording.

Open and save options

- **Open/Save from/to this PC**
  These options allow you to open a recording that is saved on your computer, or save a recording to your computer.

- **Open/Save from/to Lifecycle Services**
  This option allows you to open a recording that has been saved to a Lifecycle Services library, or save a recording to a Lifecycle Services library.

- **Open from recents**
  This option allows you to pick from a list of Task recordings that you have recently created.

- **Export as Word document**
  This option allows you to download a Word document that contains the list of steps in the recording.

- **Publish for Mobile Application**
  This option allows you to publish the task recording for a mobile application.

- **Save as developer recording**
  This option allows you to save the task recording as a developer recording.

Playback controls
Play next pending step
This option will cause Task recorder to execute and record the next pending step, which is indicated by the arrow in the Steps list.

Play to selected step
This option will cause Task recorder to begin playing and recording pending steps, beginning at the next pending step and pausing after playing the step that was selected in the list when this option was clicked.

Play all pending steps
This option will cause Task recorder to play and record all remaining pending steps, until there are no remaining pending steps.

Pause
This option only appears while playback is in progress. This option allows you to pause playback manually.

Step actions

Start sub-task/End sub-task
These options allow you to add special steps to the recording. These special steps are task steps, and you can use them to indicate when a sub-task begins and when it ends. These options are disabled while playback is in progress.

Delete step/Restore step
These options allow you to remove steps from the recording. If you delete a pending step, it will be skipped during playback, and it will not be recorded. If you delete a recorded step, then it will be flagged for removal and it will not be included in the recording when you save the recording. You can only restore steps that have been successfully recorded. You cannot delete a task while it is in progress.

Steps list
### Step counter

<table>
<thead>
<tr>
<th>5 steps recorded / 6 steps pending</th>
</tr>
</thead>
<tbody>
<tr>
<td>➔ 1 ➔</td>
</tr>
<tr>
<td>➡ 1.1 ➡</td>
</tr>
<tr>
<td>🔍 1.2 🔍</td>
</tr>
<tr>
<td>✔️ 1.3 ✔️</td>
</tr>
<tr>
<td>✔️ 1.4 ✔️</td>
</tr>
<tr>
<td>✗ ✗</td>
</tr>
</tbody>
</table>

This keeps track of how many steps have been recorded. This includes steps played by using the Playback controls, as well as steps recorded by actions that you take in the client.

### Pending step

<table>
<thead>
<tr>
<th>3 steps recorded / 9 steps pending</th>
</tr>
</thead>
<tbody>
<tr>
<td>🔍 1.8 🔍</td>
</tr>
</tbody>
</table>

This symbol represents a step that is pending and has not been recorded yet. Pending steps can be played using the Playback controls. When a pending step is played successfully, it is recorded and the symbol will update appropriately. **Pending steps are not included in the recording when you save the recording.** You must first play the pending steps so that they are recorded. If the steps are played and recorded successfully, then they will be included when you save the recording.

### Next pending step


<table>
<thead>
<tr>
<th>→ 1.2</th>
<th>Note the version number.</th>
</tr>
</thead>
</table>

This symbol represents the next pending step. If you start playback, this is the first step that will be played.

**Queued pending step**

<table>
<thead>
<tr>
<th>✗ 1.10</th>
<th>Close the page.</th>
</tr>
</thead>
</table>

This symbol represents pending steps that are queued for playback. This symbol is updated either when playback pauses, or when the queued pending step is played.

**Recorded action step**

<table>
<thead>
<tr>
<th>✔ 1.1</th>
<th>Go to About.</th>
</tr>
</thead>
</table>

This symbol represents steps that were recorded successfully, either from being played back, or manually recorded by you.

**Recorded info step**

<table>
<thead>
<tr>
<th>1.2</th>
<th>Note the version number.</th>
</tr>
</thead>
</table>

This symbol represents an info step that was played and recorded. Info steps do not result in any action being executed on the application.

**Recorded begin sub-task step**

<table>
<thead>
<tr>
<th>1</th>
<th>Sub-task: Learn about the app</th>
</tr>
</thead>
</table>

This symbol indicates the beginning of a sub-task. Sub-task steps do not result in any action being executed on the application.

**Recorded end sub-task step**

<table>
<thead>
<tr>
<th>✤ 1.11</th>
<th>End sub-task: Learn about the app</th>
</tr>
</thead>
</table>

This symbol indicates the end of a sub-task. Sub-task steps do not result in any action being executed on the application.
Deleted recorded step

This symbol represents a successfully recorded step that you have marked for deletion. Recorded steps that are marked for deletion will not be included when you save the recording. If a step has been successfully recorded when you decide to delete it, then you have the option to restore the deleted step before you save the recording.

Deleted pending step

If you delete a pending step, it will retain its pending symbol until it is played. When it is played, it will be skipped. You can restore a pending step as long as it has not been played and skipped.

Skipped step

This symbol represents a step that was deleted while it was pending, and was skipped during playback. Skipped steps are not played and are not recorded. Because skipped steps are not recorded, they are not included when you save the recording. You cannot restore a skipped step.

Error step

This symbol represents a step that was attempted by the playback system, but was not successful in being played. Error steps are not recorded, and are not included when you save the recording. You cannot restore an error step. Playback will automatically pause when an Error step is encountered. This gives you the opportunity to record replacement steps before continuing playback. An error step may occur for the following reasons:

- The step could not play because the form or lookup needed by the step was not open.
- The step could not play because the button or field needed by the step was disabled, not visible, or not present on the form.
- The step could not play because the name of the form or name of the control has changed.
- The step could not play because of a framework change to the control.
This topic explains what Task recorder and task guides are, how to create task recordings, and how to customize Microsoft task guides and include them in your Help.

**IMPORTANT**

You can record your own task guides for Dynamics 365 Human Resources, but you won't be able to save them to a Business Process Modeler (BPM) library or open them from the Help pane at this time. You can save them locally or to a network location, and then open and replays them using Task recorder.

---

**Learn about Task recorder**

Task recorder is a tool that you can use to record actions that you take in the product user interface (UI). When you use Task recorder, all of the events that you perform in the UI that are executed against the server—including adding values, changing settings, removing data—are captured. The steps that you record are collectively called a **task recording**. Task recordings can be used in many ways:

- **Task recordings can be played as task guides.** Task guides are an integral piece of the Help experience. A task guide is a controlled, guided, interactive experience through the steps of a business process. The user is instructed to complete each step by way of a pop-up prompt (or "bubble"), which will animate across the UI and point to the UI element that the user should interact with. The "bubble" also provides information about how to interact with the element, such as "Click here" or "In this field, enter a value." A task guide runs against the user’s current data set and the data that is entered is saved in the user’s environment.

- **Task recordings can be saved as Word documents.** This allows you to easily produce printable training guides.

You can create your own task recordings, play task recordings provided by Microsoft, or modify Microsoft-provided task recordings to reflect your configuration. For more information about Task recorder, see [Task recorder](#).

---

**Plan your task recording**

Whether you’re creating a new task recording or basing your recording on a Microsoft task recording, keep the following information in mind.

- Plan your recording like you would a video. Make all your decisions ahead of time.
- Walk through the business process once or twice without recording it to understand the steps.
- When you walk through the process before you record, notice where you use shortcut keys or the **Enter** key, so that you can avoid using them during the actual recording.
- Identify the following:
  - Do you want to group steps together into sub-tasks? Sub-tasks visually set apart sections of a process. For example, if you are creating a recording for "Creating and releasing a product," you may want to group together the steps that are required to create a product, and then group together the steps that are required to release the product. Sub-tasks also make longer processes easier to read.
Do you want to add annotations, and if so, where? See "Understand the different types of annotations" below for more information.

What values will you add in the various fields as you complete the steps of the business process? It is a good idea to know what you’ll select or enter as you proceed so that you don’t backtrack or fix mistakes as you’re recording.

**Write your description and annotations ahead of time**

- At the beginning of each task recording, there’s a description field that allows you to enter an introduction to the recording. It is a good idea to write and save the description ahead of time in a separate document so you can copy and paste it into the task recording when you are recording. That way, you can spend time refining the text when you aren’t in the process of recording. Cutting and pasting the text makes the recording process go more quickly and smoothly.
- For each step in a task recording, you can create annotations. During playback of a task guide, annotations appear in the “bubble” as notes above or below the text for the step. When viewed as text in the Help pane, annotations appear as text inline in the step. As with the description, it is a good idea to write and save your annotations in a separate document. When you’re recording the task recording, cut and paste the annotations in from that document.

**Understand the different types of annotations** All annotations are optional. Only add them when they’ll provide helpful information to the user.

- **Title**: A title annotation will appear before the step text that task recorder automatically generates. In the task guide, the title annotation appears above the automatically generated text. Use this type of annotation to explain why the user is doing the step or to give additional context.

This is the editing pane that you see when you add an annotation as you create your recording. Enter a title annotation in the **Title** box.
This is what the title annotation looks like in the "bubble" in the task guide.

<table>
<thead>
<tr>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double-check the information before proceeding. You are about to commit the new product to the system.</td>
</tr>
</tbody>
</table>
- **Notes**: A notes annotation will appear after the step text that task recorder automatically generates. In the task guide it will only be visible if the user clicks the *Show more* link in the task guide bubble. Use this type of annotation to describe anything that a user needs to know to complete the step.

This is the editing pane that you see when you add an annotation as you create your recording. Enter a notes annotation in the *Notes* box.
This is what the notes annotation looks like in the "bubble" in the task guide.
- **Info step:** These annotations are created by right clicking on a control or anywhere on a form in Task recorder. Info steps appear as a numbered step at whatever point you insert it, even though no action was recorded in the UI. You can add a form-level info step or an info step associated with a control. When an info step is associated with a form, the task guide “bubble” will appear someplace on the form, with no pointer, when the task guide is played. When an info step is associated with a control, the task guide “bubble” will point to the control when the task guide is played. In the Help pane, an info step annotation will appear as a numbered step with whatever text you entered. Use info steps to prepare the user for the next steps, to describe steps that need to be done outside of the application, or to refer to other recordings (although you cannot create hyperlinks in annotations).

**Determine how long to make your recording**

- The user will generally either read or play the recording from start to finish, so don’t combine steps or tasks that are better done separately.
- Try not to record a long scenario that spans multiple sub-processes. For example, “Operate in-store customer service desk” is too broad; break it up into shorter tasks such as “Accept returns” and “Add to gift card.”
- If a task can be carried out as part of several different business processes, create a separate recording for it, and you can refer to it in the other recordings.
- If the process involves multiple tasks that the person likely does all at once, you can keep the tasks in one
recording, for example, “Set up and assign functionality profiles.”

- If it is something someone does once (such as configuration) and then another task that they can do immediately afterward but may do repeatedly, and on its own, break them up into two task recordings.

**Decide where, in the UI, to start a recording** The page that you are on when you start recording a task recording affects which page the task guide is displayed for. For example, if you want your task recording to be listed in the Help pane when a user clicks Help on the General ledger parameters page, you must start your recording on the General ledger parameters page. **Save recordings as .axtr files** When you are done creating or editing a task recording, you are presented with several options for how you want to download, or save the recording. You can download the file as a task recording package (.axtr), download it as a raw recording file (.xml), download it as a Word document, or save the file to an LCS library. It is a good idea to always save your task recording as a task recording package file (.axtr). This will help make maintenance of the file easier if procedures or annotations need to change later. If you want to download the file as a Word document, also save it as a task recording package file.

**Create your task recording**

For detailed walk-through steps, see [Task recorder resources](#).

**Copy and customize Microsoft's task recordings**

You can download and edit Microsoft's task recordings to use them for your own Help documentation or training materials. To download a Microsoft task recording, follow these steps:

1. Open Task recorder. Task recorder is located in the **Settings** menu.
2. In the Task recorder pane, click **Maintain a recording**.
3. Under **Where is the recording**, click **It is in an LCS library**.
4. Click **Select the LCS library**.
5. Select the Microsoft global library.
6. In the tree, select the business process library node that the task recording is associated with.
7. Click **OK**.
8. Click **Start**.
9. At this point, step through the recording, changing any steps as you go to re-record it. **Note:** If you only need to change the text of a recording, you can open the recording in **Edit a recording's annotations** mode, and then save it.
10. After the recording has played to the end, click **Stop** in the task recorder bar at the top of the screen.
11. Choose how you want to save the task recording.

**Additional resources**

- [Help system](#)
- [Connect the Help system](#)
- [Task Recorder](#)
- [Create Rich Help Topics with Task Recorder (external link)](#)
This topic provides a list of the help topics and other resources in Dynamics 365 Supply Chain Management.

What's new and in development
Go to the Dynamics 365 Roadmap to see what new features are released and what new features are in development.

Core concepts and tasks
Select a feature area to learn more about it.

- Asset management
- Cost accounting
- Cost management
- Inventory management
- IoT Intelligence
- Master planning
- Procurement and sourcing
- Product information management
- Production control
- Sales and marketing
- Service management
- Transportation management
- Warehouse management

Dynamics 365 Finance
For information on Dynamics 365 Finance, go to the Finance home page.

Videos
This short video summarize the new supply chain management features added to Microsoft Dynamics 365 for Finance and Operations version 8.0 (April 2018).

- Synchronize a work order between Field Service and Finance and Operations

These short videos summarize the new supply chain management features added to Microsoft Dynamics 365 for Finance and Operations, Enterprise edition 7.3 (December 2017).

- Optimization advisor
- Use warehouse template to copy configuration

These short videos summarize the new supply chain management features added to Microsoft Dynamics 365 for Finance and Operations, Enterprise edition (July 2017).

- Get started with Cost accounting
- Cost control mobile workspace
● Use Excel for cost analysis
● Approve purchase orders on a mobile device
● Visual scheduling with Gantt chart for production and batch orders

The following tech conference recordings discuss supply chain management functionality from previous versions of Finance and Operations. This functionality is now part of Dynamics 365 Supply Chain Management; the same concepts still apply, and the procedures are similar in the current version.

● **Cost management:**
  ○ Overview of Cost management

● **Master planning:**
  ○ Extend the demand forecasting functionality
  ○ Master planning - tips and tricks for troubleshooting performance
  ○ MRP performance tuning

● **Product information management:**
  ○ Product configurator in Microsoft Dynamics AX

● **Warehouse management:**
  ○ Get the best out of your warehouse management system
  ○ Dynamics AX 2012 R3: Advanced warehouse management - A day in the life of process manufacturing

● **Production control videos:**
  ○ Subcontracting operations and activities in manufacturing

● **Transportation management videos:**
  ○ Transportation management (TMS) in the new Microsoft Dynamics AX

**Blogs**

There are many topics about manufacturing and supply chain management on the Dynamics AX Manufacturing R&D Team Blog and Supply Chain Management in Dynamics AX R&D Team Blog. Most of these were written for the previous version, but the same concepts still apply, and the procedures are similar in the current version.

**White papers**

● Lean manufacturing: Capable to promise and kanban job scheduling
● BOM calculation by using a costing sheet

**eLearning courses**

For online courses and training, check out Dynamics 365 Supply Chain Management on Microsoft Learn.
This topic provides a list of the help topics and other resources for the financial management features in Microsoft Dynamics 365 Finance.

Select a feature area to learn more about it.

- Accounts payable
- Accounts receivable
- Asset leasing
- Budgeting
- Cash and bank management
- Cost accounting
- Financial reporting
- Fixed assets
- Finance insights
- General ledger and Financial reporting
- Public sector

Additional resources

Blogs
- Microsoft Dynamics 365 blog
- Financials blog
- Microsoft Dynamics Operations Partner Community Blog

Videos
Check out the how-to videos that are now available on the Microsoft Dynamics 365 YouTube Channel.

Country/region functionality
Country/region regulations affect tax setup and other areas of financial management. To learn about country/region-specific functionality, see Localization and regulatory features.

Additional content
Supply chain management functionality covers parts of the procure-to-pay process that include:

- Requisitioning
- Ordering
- Receiving
- Invoicing
- Paying for the goods and services your organization purchases

For information about the capabilities for managing purchases, inventory, and manufacturing, see the Supply Chain Management home page.

eLearning courses
For online courses and training, check out Dynamics 365 Finance on Microsoft Learn.
Dynamics 365 Commerce—built on the proven Dynamics 365 Retail capabilities—delivers a comprehensive omnichannel solution that unifies back-office, in-store, call center, and digital experiences. Dynamics 365 Commerce enables you to build brand loyalty through personalized customer engagements, increase revenue with improved employee productivity, optimize operations to reduce costs and drive supply chain efficiencies, ultimately delivering better business outcomes.

This release enables the creation of digital experiences using built-in web authoring and development tools to produce engaging and intelligent digital storefronts. A connected marketing and headless commerce platform further enable the seamless management of content, assets, promotions, inventory, and pricing across all channels.

- **Everything to build and run digital commerce**: Streamline your business and end-to-end commerce solution that scales to your needs across traditional and emerging channels. Built-in web authoring and development tools enable you to create engaging intelligent digital storefronts, while a connected marketing and headless commerce platform enables seamless management of content, assets, promotions, inventory, and pricing across channels.

- **Build loyalty and exceed customer expectations**: Use clienteling tools to gain a comprehensive view of your customer and respond to their needs at every level of engagement, based on customer profile, history, and preferences that flow across physical and digital channels. Empower your employees to foster lasting relationships through AI-driven recommendations, customer insights, and loyalty programs that elevate brand appeal.

- **Flexible and intelligent omnichannel experience**: Unify physical and digital commerce by providing consistent experiences to customers across cloud search and discovery, product reviews, wish lists, inventory, gift cards, and loyalty. Allow customers to purchase when, how, and where they want, on any device—while providing choice around modern payment methods and product collection or delivery.

- **Streamline operations using AI in the cloud**: Drive omnichannel commerce experiences and integrated, optimized back-office operations through ingrained, pervasive, and context-aware cloud intelligence. Use advanced merchandising, inventory management, distributed order management, and pricing and promotion to innovate and stay ahead of competition. Derive insights by visualizing and analyzing comprehensive and consistent data across all aspects of your business. Use AI-driven technologies to provide accessible websites, protect your business against payment fraud, and efficiently moderate user-generated content like ratings and reviews.

### Core concepts and tasks

Select a feature area to learn more about it:

- Configure a Commerce preview environment
- Commerce architecture
- Set up your channels
- Merchandizing your products and services
- Manage your orders
- Manage your customers
- Manage your financials
● Manage your e-Commerce site
● Fraud protection
● Commerce development and extensibility

eLearning courses

For online courses and training, check out Dynamics 365 Commerce on Microsoft Learn.